Author/s:
Avanzi, B;Dong, ET;Laub, PJ;Wong, B

Title:
Distributional refinement network: Distributional forecasting via deep learning

Date:
2026-05-01

Citation:
Avanzi, B., Dong, E. T., Laub, P. J. & Wong, B. (2026). Distributional refinement network: Distributional forecasting via deep learning. Insurance: Mathematics and Economics, 128, https://doi.org/10.1016/j.insmatheco.2026.103246.

Persistent Link:
https://hdl.handle.net/11343/370660

Insurance Mathematics and Economics 128 (2026) 103246

Contents lists available at ScienceDirect

Insurance: Mathematics and Economics

journal homepage: www.elsevier.com/locate/ime

Distributional refinement network: Distributional forecasting via deep learning

Benjamin Avanzi [a], Eric T. Dong [b,*], Patrick J. Laub [b], Bernard Wong [b]

[a] Centre for Actuarial Studies, Department of Economics, University of Melbourne VIC 3010, Australia
[b] School of Risk and Actuarial Studies, UNSW Business School, UNSW Sydney NSW 2052, Australia

ARTICLE INFO

JEL classification:
C51
C53
G22

2000 MSC:
91G70
91G60
62P05
91B30

Keywords:
Insurance pricing
Deep learning
Distributional regression
Probabilistic forecasting
Model interpretability

ABSTRACT

A key task in actuarial modelling involves modelling the distributional properties of losses. Classic (distributional) regression approaches like Generalised Linear Models (GLMs) are commonly used, but challenges remain in developing models that can (i) allow covariates to flexibly impact different aspects of the conditional distribution, (ii) integrate developments in machine learning and AI to maximise the predictive power while considering (i), and, (iii) maintain a level of interpretability in the model to enhance trust in the model and its outputs, which is often compromised in efforts pursuing (i) and (ii).

We tackle this problem by proposing a Distributional Refinement Network (DRN), which combines an inherently interpretable baseline model (such as GLMs) with a flexible neural network–a modified Deep Distribution Regression (DDR) method. Specifically, our approach flexibly refines the baseline distribution. As a result, the DRN captures varying effects of features across all quantiles, improving predictive performance while maintaining adequate interpretability.

Using both synthetic and real-world data, we demonstrate the DRN's superior distributional forecasting capacity. The DRN has the potential to be a powerful distributional regression model in actuarial science and beyond.

1. Introduction

1.1. Background and motivation

In regression problems within general insurance, an important objective is to model the distributional properties of losses, such as the mean, variance, and quantiles, given the risk features (Frees et al., 2014; Embrechts and Wüthrich, 2022). An associated significant challenge, also recognised in statistics and machine learning (Kneib et al., 2021; Klein, 2024), involves developing models that accurately capture the entire distribution of the response variable based on its features (or covariates, explanatory variables). This requires models with substantial distributional flexibility–precisely capturing the varying impacts of features across all quantiles of the response variable's conditional distribution (Li et al., 2021; Fissler et al., 2023). Overcoming this challenge improves forecasting precision concerning key distributional properties, which is vital for informed decision-making within the actuarial context (Frees, 2009). Deep learning techniques have gained attention for their potential to empower models with greater distributional flexibility (Bishop, 1994; Taylor, 2000; Li et al., 2021). However, the additional flexibility provided by neural network adaptations often comes at the cost of "indispensable" interpretability (Rügamer et al., 2023); see also Richman (2022) and Embrechts and Wüthrich (2022).

Balancing flexibility and interpretability remains an ongoing challenge. Traditional parametric distributional regression models, such as Generalised Linear Models (GLMs; Nelder and Wedderburn, 1972), offer an interpretable and structured framework but are often limited by their rigid distributional assumptions. These models typically consider only Exponential Family Distributions (EFDs), where variance and quantiles are deterministic functions of the mean, influenced by the treatment of dispersion. This restricts their ability to capture the diverse impacts of features on key distributional properties, which is crucial in fields like insurance pricing (Delong et al., 2021; Fung et al., 2022) and loss reserving (Al-Mudafer et al., 2022). More complex parametric models, such as

* Corresponding author.
E-mail addresses: b.avanzi@unimelb.edu.au (B. Avanzi), tian.dong@unsw.edu.au (E.T. Dong), p.laub@unsw.edu.au (P.J. Laub), bernard.wong@unsw.edu.au (B. Wong).

https://doi.org/10.1016/j.insmatheco.2026.103246
Received 3 June 2024; Received in revised form 29 November 2025; Accepted 24 March 2026
Available online 29 March 2026

deep feedforward neural networks with distributional assumptions and Mixture Density Networks (MDNs; Bishop, 1994), improve flexibility but often lack interpretability and are prone to overfitting (Makansi et al., 2019). Semiparametric methods, like quantile regression (Koenker and Bassett, 1978) and distribution regression (Foresi and Peracchi, 1995), enhance distributional flexibility by relaxing distributional assumptions. However, as argued by Zeng and Lin (2007), they often lack efficient estimators and the rigorous theoretical frameworks found in parametric models. They can also adopt unrealistic structural assumptions, leading to issues like quantile crossings in quantile regression (Klein, 2024). Deep learning adaptations of the above methods, such as deep quantile regression (Taylor, 2000) and Deep Distribution Regression (DDR; Li et al., 2021), further lack the inherent distributional interpretability. The Neural Additive Models for Location Scale and Shape (NAMLSS; Thielmann et al., 2024) architecture is designed to combine interpretability, flexibility, and neural networks, but has a limited capacity to capture interaction effects which need to be handled manually (see, e.g., Laub et al., 2025). For high-stakes decision-making, inherently interpretable models like GLMs are often preferred over post-hoc interpretability techniques such as Local Interpretable Model-agnostic Explanations (LIME; Ribeiro et al., 2016) and Shapley Additive Explanations (SHAP; Lundberg and Lee, 2017), which provide explanations only after predictions have been made. They often rely on the unrealistic assumption that features are independent of one another, leading to misleading interpretations of model behaviour (Jethani et al., 2022; Mayer et al., 2023). As a result, integrating traditional, inherently interpretable GLM frameworks within neural network structures has been proposed, as seen in models like DeepGLM (Tran et al., 2020), Combined Actuarial Neural Network (CANN; Schelldorfer and Wüthrich, 2019), and LocalGLMnet (Richman and Wüthrich, 2022). However, these parametric frameworks still face the issue of limited distributional flexibility, focusing primarily on the mean and falling short of the ultimate goal of distributional regression (Kneib et al., 2021; Klein, 2024).

1.2. Contributions

A major challenge associated with distributional regression methods such as the DDR, which maximise distributional flexibility to accurately predict the entire conditional distribution, is with preserving sufficient distributional interpretability. To address this challenge, we introduce the Distributional Refinement Network (DRN), a deep learning framework that provides exceptional distributional flexibility while maintaining a level of interpretability. Inspired by baseline-refining models in the actuarial literature, such as CANN, the DRN integrates an inherently interpretable parametric model, such as a GLM, within a flexible semiparametric deep learning framework inspired by the DDR. It can be regarded as a deep-learning-refined version of a GLM or other interpretable parametric distributional regression models. While the GLM component provides initial estimates of predicted densities, the neural network component then refines these estimates in a regularised manner by utilising the available data to minimise a loss function focused on overall distributional forecasting performance.

Fig. 1 provides a schematic representation of the DRN framework. The key benefits of our approach are as follows:

1. The DRN achieves high distributional flexibility by leveraging neural networks to adjust various aspects of the response variable's conditional distribution. Unlike DDR, which directly learns the full distribution on a finite range, DRN refines a reliable baseline distributional regression model, using it as a well-informed starting point. Additionally, DRN introduces a unique partitioning method that refines the baseline at a finer resolution. This allows for greater precision in specific regions where necessary instead of the full region as in the DDR case. This improved distributional flexibility enhances predictive performance, addressing a primary goal in distributional regression within actuarial contexts and beyond.
2. The DRN integrates an interpretable baseline model and incorporates tailored regularisation techniques. Specifically, it applies KL divergence to penalise deviations from the baseline in a distributional sense and enforces a roughness penalty to smooth the distribution. These mechanisms combat overfitting and preserve distributional transparency–an advantage over flexible distributional regression approaches like DDR, which lack a built-in reference structure. By explicitly anchoring a black-box model to a well-trusted baseline, DRN ensures that a significant portion of the prediction remains interpretable and closely tied to a credible reference point, which makes it more reliable. This structure opens up possibilities for assessing how refined distributions deviate from the baseline, offering valuable insights into potential adjustments if desired.

1.3. Paper structure

The remainder of this paper is organised as follows. Section 2 introduces the general setting and notation used in the paper. Section 3 reviews existing distributional forecasting models. Section 4 introduces the Distributional Refinement Network. Section 5 examines the performance of the proposed DRN compared to existing approaches using a synthetic dataset. Section 6 validates the DRN's ability to enhance distributional forecasts with a real-world claim severity dataset. Section 7 concludes.

2. General setting and notation

Our general setting and associated notation can be summarised as follows:

- $\boldsymbol{x}_i = (x_{i,1}, \dots, x_{i,p})^\top \in \mathbb{R}^p$ corresponds to the features for the ith observation, $y_i \in \mathbb{R}$ represents the response variable for the ith observation, and p represents the dimension of the features. The complete dataset is denoted by $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, with the training, validation and test datasets represented as $\mathcal{D}_{\text{Train}}$, $\mathcal{D}_{\text{Val}}$, and $\mathcal{D}_{\text{Test}}$, respectively.
- The random variable $Y|\boldsymbol{X}$ represents the response variable Y, conditioned on the explanatory variables $\boldsymbol{X}$. The random variables Y_i's given $\boldsymbol{x}_i$'s are independent. The functions $f_{Y|\boldsymbol{X}}(y|\boldsymbol{x};\boldsymbol{w})$ and $F_{Y|\boldsymbol{X}}(y|\boldsymbol{x};\boldsymbol{w})$ represent the conditional probability density and cumulative distribution function, respectively, of the response variable Y given the features $\boldsymbol{x}$ and model parameters/weights $\boldsymbol{w}$. The symbols $\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{X}]$ and $\mathbb{V}_{Y|\boldsymbol{X}}[Y|\boldsymbol{X}]$ represent the conditional mean and variance, respectively, of $Y|\boldsymbol{X}$. Additionally, $Q_{Y|\boldsymbol{X}}(\alpha|\boldsymbol{X})$ signifies the α quantile of $Y|\boldsymbol{X}$.
- The quantile function, for a specific instance $\boldsymbol{x}$ and model parameters $\boldsymbol{w}$, computes the α quantile. In the context of actuarial science and finance, it corresponds to the Value at Risk (VaR) at the $1-\alpha$ level, denoted as $\text{VaR}_{1-\alpha}(Y|\boldsymbol{x};\boldsymbol{w})$. We adopt analogous notation for the predicted conditional mean and variance for a given instance.
- For the implementation of a specific distributional regression technique referred to as DF, we define the model $\mathcal{M}_{\boldsymbol{w}_{\text{DF}}}$ by its trained parameters or weights represented as $\boldsymbol{w}_{\text{DF}}$. When employing a GLM, we set $\boldsymbol{w}_{\text{GLM}} = \boldsymbol{\beta}_{\text{GLM}} = \boldsymbol{\beta}$, conforming to the widely accepted notation in existing literature. Likewise, we can abbreviate the notation $\mathcal{M}_{\boldsymbol{w}_{\text{GLM}}}$ to $\mu_{\boldsymbol{\beta}}$ for simplicity. Here, $\mu_{\boldsymbol{\beta}}$ represents the mean regression function that is parameterised by $\boldsymbol{\beta}$.
- The conditional distribution of $Y|\boldsymbol{X}$ is assumed to follow a specific distribution $\mathcal{F}$, characterised by K parameters, denoted as $\boldsymbol{\theta}(\boldsymbol{X}) = (\theta_1(\boldsymbol{X}), \dots, \theta_K(\boldsymbol{X}))^\top$. When distributional parameters are estimated using a common set of weights, we use $\theta_k(\boldsymbol{X};\boldsymbol{w}_{\text{DF}})$ to denote the kth distributional parameter estimated by the DF approach. Conversely, when unique weights are assigned to predict each distributional parameter, $\boldsymbol{w}_{\text{DF},k}$ represents the weights specific to the kth distributional parameter.

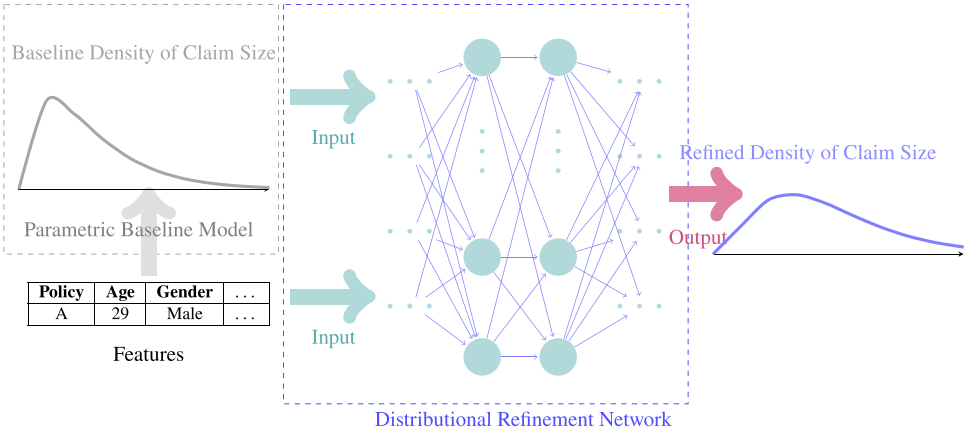


Fig. 1. A schematic view of our proposed deep learning framework designed for distributional forecasting. The Distributional Refinement Network (DRN) inputs a feature set and a density estimate provided by a parametric baseline model, such as a GLM. The DRN then outputs a refined version of the initial density estimate.

3. Existing distributional regression techniques

This section offers a literature review of the current methods for distributional regression, broadly categorised into parametric and semi-parametric approaches. Parametric models such as the Generalised Linear Model (GLM), Generalised Additive Models for Location, Scale, and Shape (GAMLSS), Combined Actuarial Neural Network (CANN), and Mixture Density Network (MDN) embed specific distributional assumptions within their frameworks. Traditional actuarial modelling often relies on the former two models, while the latter two have been extensively explored for their predictive capabilities. Conversely, semiparametric models offer enhanced adaptability and flexibility by avoiding imposing assumptions on the distributions of the response variable conditional on its features. Notable examples include quantile regression and distribution regression. Nonetheless, the deep-learning extensions of these models are yet to be explored and effectively implemented for actuarial applications.

Among these methods, CANN and Deep Distribution Regression (DDR) are particularly relevant to our proposed Distributional Refinement Network (DRN). The following subsections detail these methodologies, while a summary of the well-established GLM, GAMLSS, a simple feedforward neural network, MDN and deep quantile regression is provided in Appendix A.

3.1. Combined actuarial neural network (CANN)

Researchers have explored the combination of GLMs with simple feedforward neural networks, as highlighted by Tran et al. (2020), to leverage both the favourable statistical properties of GLMs and the rich flexibility of neural networks. In actuarial science, Schelldorfer and Wüthrich (2019) proposed the nesting of the GLM in a neural network known as the Combined Actuarial Neural Network (CANN) $\mathcal{M}_{\boldsymbol{w}_{\text{CANN}}} : \mathbb{R}^p \to \mathbb{R}$:

$$\mathcal{M}_{\boldsymbol{w}_{\text{CANN}}}(\boldsymbol{x}) = g^{-1}\left(\left(\beta_0 + \sum_{j=1}^{p} \beta_j x_j\right) + a^{(L,L+1)}(\boldsymbol{z}^{(L)})\right), \tag{1}$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^\top \in \mathbb{R}^{p+1}$ is the vector of regression coefficients obtained through the GLM specified in Section A.1. The loss function for training the CANN inherits the deviance loss function adopted for training the baseline GLM.

3.2. Deep distribution regression (DDR)

Recently, Li et al. (2021) developed Deep Distribution Regression (DDR), extending the principles of traditional distribution regression to deep learning. Distribution regression is a statistical technique that models the conditional distribution of the response variable given features without making distributional assumptions. Foresi and Peracchi (1995) proposed using the generalised additive model to estimate the conditional distributions directly, and Wang et al. (2023) extended the technique for use in traditional actuarial science problems. Building on these foundations, DDR trains a neural network to directly estimate the conditional density.

One of the key steps is partitioning the support of the response variable. This partitioning step is crucial for several reasons: i) tackling the challenges of specifying the architectures and nature of outputs; ii) the concept of fitting distributional parameters that determine the density levels for the entire range of the response variable is no longer valid due to the distribution-free nature regarding the response variable; iii) enhanced training tractability and efficiency since the conditional density estimation task is converted into a classification problem easier for neural networks to handle; iv) improved practicability with the implementation of loss functions, such as NLL and joint binary cross-entropy (JBCE) defined in Eq. (3). The method of partitioning is as follows:

1. Identify an overall interval of focus (c_0, c_K) of the response variable Y for adjustment, where $c_K = u \in \mathbb{R}$ is the upper bound and $c_0 = l \in \mathbb{R}$ is the lower bound.
2. Partition the interval into K non-overlapping bins by specifying the cutpoints $c_1, c_2, \ldots, c_{K-1}$. The kth partitioned interval, denoted as $T_k = [c_k, c_{k+1})$, has equal width $|T_k| = c_{k+1} - c_k$ for all $k \in \{0, 1, \ldots, K-1\}$.

After specifying K partitioned intervals, the DDR network $\mathcal{M}_{\boldsymbol{w}_{\text{DDR}}} : \mathbb{R}^p \to \mathbb{R}^K$ follows

$$\mathcal{M}_{\boldsymbol{w}_{\text{DDR}}}(\boldsymbol{x}) = \left(\pi_1\left(\boldsymbol{x}; \boldsymbol{w}_{\text{DDR}}\right), \ldots, \pi_K\left(\boldsymbol{x}; \boldsymbol{w}_{\text{DDR}}\right)\right)^\top, \tag{2}$$

where $\pi_k(\boldsymbol{x}; \boldsymbol{w}_{\text{DDR}}) = \mathbb{P}(Y \in T_k | \boldsymbol{x}; \boldsymbol{w}_{\text{DDR}})$. The loss function is given by the joint binary cross-entropy (JBCE) loss function:

$$\begin{aligned}\mathcal{L}_{\text{DDR}}(\boldsymbol{w}, \mathcal{D}) = -\sum_{k=1}^{K}\sum_{i=1}^{N}\Big[&\mathbb{1}_{\{y_i \le c_k\}} \ln F_{Y|\boldsymbol{X}}(c_k|\boldsymbol{x}_i; \boldsymbol{w}) \\ &+ \left(1 - \mathbb{1}_{\{y_i \le c_k\}}\right) \ln\left(1 - F_{Y|\boldsymbol{X}}(c_k|\boldsymbol{x}_i; \boldsymbol{w})\right)\Big].\end{aligned} \tag{3}$$

Intuitively, the JBCE loss penalises the discrepancies between the predicted cumulative distribution function (CDF) and actual distribution at various cutpoints, encouraging the model to produce accurate CDF estimates. The DDR model estimates density as a piecewise constant, as illustrated in Fig. 2.

4. Distributional refinement network

This section presents the Distributional Refinement Network (DRN) framework, designed for distributional forecasting within actuarial sci-

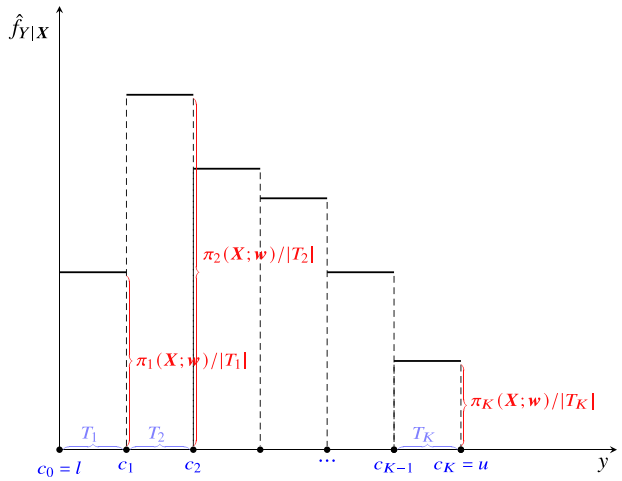


Fig. 2. Demonstration of the DDR model proposed by Li et al. (2021).

ence and applicable to broader contexts. The DRN is a feedforward neural network that refines an interpretable baseline model's predicted density functions. The underlying motivation is that the DRN offers sufficient flexibility to fine-tune baseline densities such that the refined densities better match the true data-generating density functions. Such an approach can relax the baseline model's rigid distributional or functional constraints, enhancing distributional flexibility by empowering the neural networks to learn the diverse impact of features on all quantiles. Integrating an inherently interpretable baseline model ensures distributional interpretability, rendering it applicable to diverse applications.

Inputs to the DRN comprise both features and the transformed density functions predicted by the baseline distributional regression model. The related integration procedures and the DRN's architecture are detailed in Section 4.1. Section 4.2 explains how the DRN generates and evaluates its distributional forecasts, with details on tailored regularisation for generalisation and robustness considerations.

Without loss of generality, but for notational clarity and convenience, and to agree with a typical actuarial setting, we adopt $\boldsymbol{\beta}$ to denote the baseline model weights $\boldsymbol{w}_{\text{Baseline}}$, and $\boldsymbol{w}$ to denote the DRN weights $\boldsymbol{w}_{\text{DRN}}$. Accordingly, the conditional density functions estimated by the baseline model and the DRN are represented by $f_{Y|\boldsymbol{X}}(y|\boldsymbol{x};\boldsymbol{\beta})$ and $f_{Y|\boldsymbol{X}}(y|\boldsymbol{x};\boldsymbol{w},\boldsymbol{\beta})$, respectively. This notation convention extends similarly to the estimated cumulative distribution functions (CDFs) $F_{Y|\boldsymbol{X}}$'s.

4.1. Architecture design

This section delves into the DRN's comprehensive structural design and construction. Section 4.1.1 covers the necessary steps to integrate a baseline model's distributional forecasting information in the DRN. The specific architecture of the DRN and its method for generating outputs are elaborated in Section 4.1.2.

4.1.1. Baseline model selection and integration

The distinctive characteristic of our proposed Distributional Refinement Network (DRN) is its ability to incorporate the forecasting insights from a baseline distributional regression model into its training process. The baseline model greatly impacts the DRN by providing essential "prior" distributional forecasting information. This setup enables the users to modulate the contributions between the baseline and the neural network components. This balance is regulated by a Kullback–Leibler (KL) divergence (Kullback and Leibler, 1951) penalty term, which is further elaborated in Section 4.2.2. Essentially, the DRN's distributional forecasting performance relies on both the initialisation of the neural network's weights and the parameter estimates derived from the baseline model. The choice of regularisation adds further complexity, as even the direction of KL divergence can significantly impact results when intertwined with baseline selection. See Remark 1 and Appendix D for a detailed discussion.

Therefore, the selection of an appropriate baseline model becomes a nuanced decision pivotal to the DRN's performance. An ideal baseline model is expected to yield a reasonable distributional forecast across a given feature set $\boldsymbol{x}$. This involves generating both the PDF, $f_{Y|\boldsymbol{X}}(y|\boldsymbol{x};\boldsymbol{\beta})$, and the CDF, $F_{Y|\boldsymbol{X}}(y|\boldsymbol{x};\boldsymbol{\beta})$. From a philosophical perspective, the baseline model should also be inherently interpretable. Regaining interpretability might be more challenging for a complex baseline model, and such a model may not be optimal for a straightforward problem or limited data. To this end, two natural candidates are GLM and GAMLSS. For distributional interpretability considerations, especially regarding the mean and occasionally the variance, GLM typically outperforms GAMLSS. Nonetheless, a GAMLSS baseline might offer improved distributional forecasting performance due to higher flexibility. Ultimately, the decision rests on the context of the problem and the required level of interpretability for the model. Throughout the paper, we select GLM as the baseline model because of its exceptional distributional interpretability and wide acceptance in actuarial applications. The impact of baseline choice, along with recommended strategies, is detailed in Section 5.4 and Appendix D.

Incorporating a baseline model into a feedforward neural network is challenging. A feedforward neural network cannot directly process continuous density functions $f_{Y|\boldsymbol{X}}(y|\boldsymbol{x};\boldsymbol{\beta})$'s, which are defined through mathematical relationships predicted by the baseline model. Transforming the baseline density function into a numerical format the DRN can process is essential. One solution is to partition the density function into non-overlapping segments, converting information into a format compatible with the DRN. Specifically, inspired by the DDR methodology, we introduce a "Partitioned Piecewise Constant" (PPC) transformation that employs the partitioning strategy outlined in Section 3.2. This transformation is designed to modify the baseline density function, $f_{Y|\boldsymbol{X}}(y|\boldsymbol{x};\boldsymbol{\beta})$, into a piecewise constant function across specified intervals determined by a series of predefined cutpoints $c_0 < c_1 < \ldots < c_{K-1} < c_K$. The length of the kth interval is given by $|T_k| = c_k - c_{k-1}$ for $k \in \{1, \ldots, K\}$. The assumption is that the density function remains constant within each partitioned interval:

$$\text{PPC}(f_{Y|\boldsymbol{X}}(y|\boldsymbol{x};\boldsymbol{\beta})) = \begin{cases} b_k(\boldsymbol{x};\boldsymbol{\beta})/|T_k|, & \text{if } y \in [c_{k-1}, c_k) \text{ for } k = 1, \ldots, K, \\ f_{Y|\boldsymbol{X}}(y|\boldsymbol{x};\boldsymbol{\beta}), & \text{otherwise,} \end{cases} \tag{4}$$

where $b_k(\boldsymbol{x};\boldsymbol{\beta})$ calculates the predicted baseline probability mass for the kth partitioned interval. This is defined by the difference in the CDF values at the cutpoints, i.e.,

$$b_k(\boldsymbol{x};\boldsymbol{\beta}) = F_{Y|\boldsymbol{X}}(c_k|\boldsymbol{x};\boldsymbol{\beta}) - F_{Y|\boldsymbol{X}}(c_{k-1}|\boldsymbol{x};\boldsymbol{\beta}). \tag{5}$$

The baseline probability masses $b_k(\boldsymbol{x};\boldsymbol{\beta})$'s are utilised as inputs into the DRN, a process which is thoroughly detailed in Section 4.1.2. Hence, identifying suitable partitioning cutpoints $c_0, \ldots, c_K$, as described in (4), becomes a strategic consideration to transform baseline density for the DRN; see also Section 3.2. A simple partitioning method involves selecting lower and upper bounds, denoted as c_0^* and c_K^*, respectively. These are typically set to values slightly below the minimum and slightly above the maximum values of the response variable in the training dataset. Subsequently, we establish a uniform grid of cutpoints within these bounds. This naive uniform partitioning method presents a straightforward initial approach; it is easily customised. An improvement could involve recalibrating cutpoints to ensure that each interval contains at least $M \in \mathbb{Z}^+$ observations from the training dataset. Details on the algorithm used to merge cutpoints can be found in Appendix B. In practice, as observed in both our work and that of Li et al. (2021), setting $K = 0.1 \times |\mathcal{D}_{\text{Train}}|$, i.e., a cutpoint-to-training observations ratio of approximately 0.1, tends to perform reasonably well. There are certain scenarios where increasing this ratio may be beneficial: (i) when the training dataset

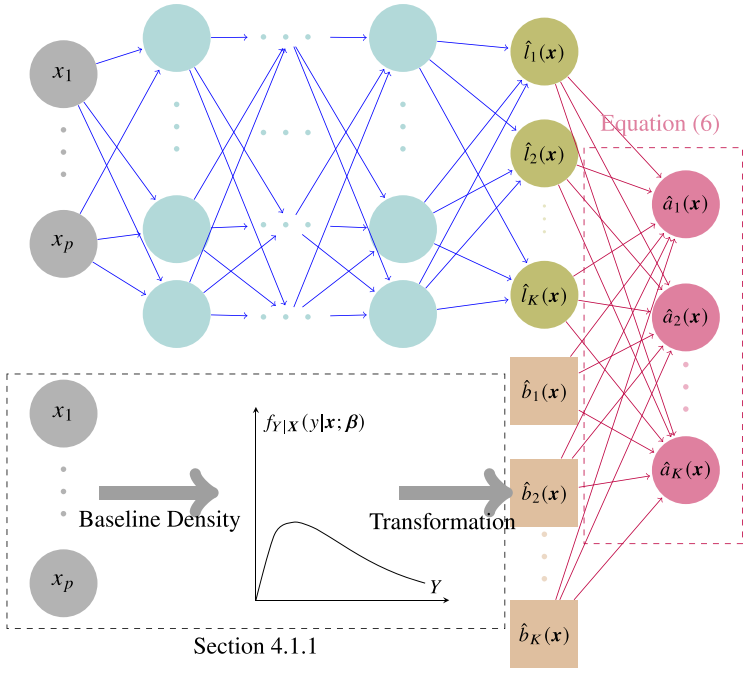


Fig. 3. The schematic illustrates the architecture of the Distributional Refinement Network (DRN). Notably, the baseline model depicted in the bottom left is not directly fed into the network. Instead, the DRN employs the summarised baseline information, $\boldsymbol{b}(\boldsymbol{x};\boldsymbol{\beta}) = (\hat{b}_1(\boldsymbol{x}), \dots, \hat{b}_K(\boldsymbol{x}))^\top$, which comprises the predicted baseline probability masses as outlined in Eq. (5). Represented by the brown square-shaped neurons, this information is introduced into the DRN as a skip-connection, combined with the "raw" outputs, $\boldsymbol{l}(\boldsymbol{x};\boldsymbol{w}) = (\hat{l}_1(\boldsymbol{x}), \dots, \hat{l}_K(\boldsymbol{x}))^\top$. These "raw" outputs are obtained from the feature inputs $\boldsymbol{x} = (x_1, \dots, x_p)^\top$ (top left) after propagation through the hidden layers, depicted by the blue neurons. The red arrows in the figure signify the process of computing adjustment factors $\boldsymbol{a}(\boldsymbol{x};\boldsymbol{w},\boldsymbol{\beta}) = (\hat{a}_1(\boldsymbol{x}), \dots, \hat{a}_K(\boldsymbol{x}))^\top$. They reflect the specific transformation and integration steps within the DRN, defined through Eq. (6). This process remains unchanged during the backpropagation of weights. The trainable weights within the DRN are denoted by blue lines, while the grey arrows denote a series of transformations elaborated in Section 4.1.1. The network employs `LeakyReLU` for hidden layers to prevent inactive neuron issues and `Linear` for the output layer. The detailed explanations of these choices are provided in Appendix E. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

is very limited, and (ii) when the response variable exhibits heavy-tailedness. Both cases can lead to an extremely wide gap between c_0^* and c_K^*, which limits NN's flexibility to adjust the probability distribution in specific regions. Conversely, we find that decreasing the cutpoint-to-observation ratio can be sometimes advantageous when training data is abundant. This reduces the risk of overfitting by avoiding overly frequent or unnecessarily granular cutpoints, which is the opposite logic of the scenarios described above. The minimum number of observations per interval is found to (i) allow for larger values when ample training data is available, and (ii) have a generally less impactful effect on distributional performance. In practice, values such as 1, 3, or 5 can typically be used as starting points. For optimal performance, we recommend tuning both the cutpoint-to-observation ratio and the minimum observations per interval. Nevertheless, the above discussion provides a reasonable and practical range for initial hyperparameter selection. Apparently, with prior knowledge, it is possible to manually pick the partitioning cutpoints.

4.1.2. Output generation

Fig. 3 illustrates the architecture of the DRN, structured as a feed-forward neural network, with blue arrows representing the trainable weights. Inputs to the neural network include the feature set $\boldsymbol{x}$ along with the baseline model's probability masses $b_k(\boldsymbol{x};\boldsymbol{\beta})$'s as defined in Eq. (5). These probability masses are depicted by neurons coloured brown. The DRN's "raw" outputs $(l_1(\boldsymbol{x};\boldsymbol{w}), \dots, l_K(\boldsymbol{x};\boldsymbol{w}))^\top \in \mathbb{R}^K$, are represented by olive-coloured neurons. Each of these outputs, $l_k(\boldsymbol{x};\boldsymbol{w})$, is a real-valued function crucial for the process of generating adjustment factors.

The "raw" outputs $l_k(\boldsymbol{x};\boldsymbol{w})$'s are combined with the baseline model's probability masses $b_k(\boldsymbol{x};\boldsymbol{\beta})$'s to produce adjustment factors $a_k(\boldsymbol{x};\boldsymbol{w},\boldsymbol{\beta})$'s

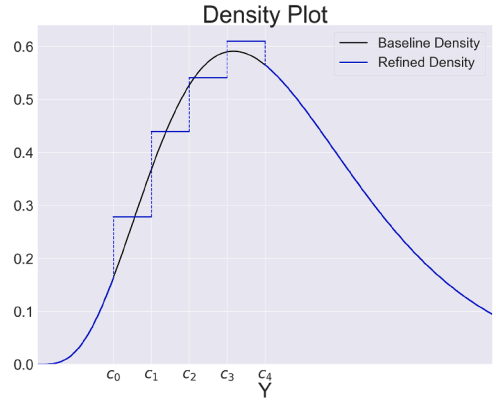


Fig. 4. The figure showcases the adjustment factors $\hat{a}_k$'s applied to the density of $Y|\boldsymbol{X} = \boldsymbol{x}$, with dashed vertical lines marking the cutpoints, c_k's. For illustration, five cutpoints $c_0 < c_1 < c_2 < c_3 < c_4$ form four intervals that require density modifications. The black line denotes the baseline model's density, while the blue line indicates the refined density as estimated by the DRN. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

for the transformed baseline density functions. The adjustment factor for each partitioned interval is defined in the formula below, with its procedural depiction in Fig. 3 represented by red arrows, illustrating the specific transformation step involved:

$$\hat{a}_k(\boldsymbol{x}) = a_k(\boldsymbol{x};\boldsymbol{w},\boldsymbol{\beta}) = \left(F_{Y|\boldsymbol{X}}(c_K|\boldsymbol{x};\boldsymbol{\beta}) - F_{Y|\boldsymbol{X}}(c_0|\boldsymbol{x};\boldsymbol{\beta})\right) \cdot \frac{\exp(l_k(\boldsymbol{x};\boldsymbol{w}))}{\sum_{j=1}^K b_j(\boldsymbol{x};\boldsymbol{\beta}) \cdot \exp(l_j(\boldsymbol{x};\boldsymbol{w}))}. \tag{6}$$

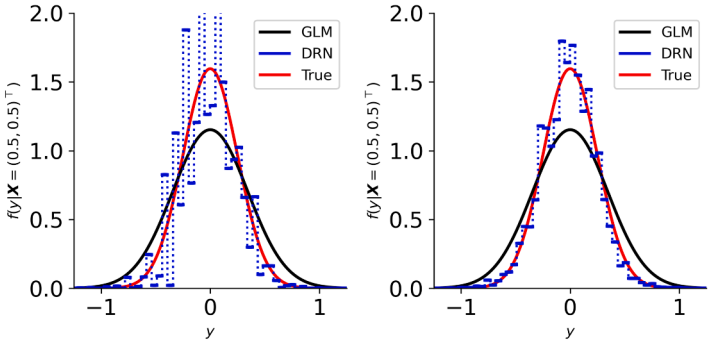


Fig. 5. The conditional densities generated by the DRN are shown with and without KL regularisation applied. The left graph shows the unregularised DRN density estimate ($\alpha_1 = 0$) and the right graph shows KL regularisation applied ($\alpha_1 = 0.001$). The DRN density function (blue) is compared against the baseline GLM (black) density and the true density (red) for the specific observation $\boldsymbol{X} = (0.5, 0.5)$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

This calculation ensures that the product of the adjustment factor and the baseline probability mass equals the refined probability mass within a specific interval, i.e.,

$$\begin{aligned}&\mathbb{P}(Y \in T_k \mid \boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\beta}) = a_k(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\beta}) \cdot b_k(\boldsymbol{x}; \boldsymbol{\beta}) \\ &= \big(F_{Y|\boldsymbol{X}}(c_K|\boldsymbol{x}; \boldsymbol{\beta}) - F_{Y|\boldsymbol{X}}(c_0|\boldsymbol{x}; \boldsymbol{\beta})\big) \cdot \\ &\text{Softmax}\big(\{\log(b_j(\boldsymbol{x}; \boldsymbol{\beta})) + l_j(\boldsymbol{x}; \boldsymbol{w})\}_{j=1,\dots,K}\big)_k, \end{aligned} \tag{7}$$

where Softmax function is defined as

$$\text{Softmax}(\boldsymbol{x})_k = \frac{\exp(x_k)}{\sum_{j=1}^{K} \exp(x_j)}. \tag{8}$$

4.2. Distributional forecasting

This section details the methodologies used in the DRN's distributional forecasting and its evaluation. Section 4.2.1 describes the process by which DRN forecasts PDFs and CDFs. Candidates for the loss function and regularisation techniques to optimise the DRN training are elaborated in Section 4.2.2. Section 4.2.3 offers a discussion of the evaluation metrics used.

4.2.1. Distribution generation

The DRN generates the density and cumulative distribution functions forecasts utilising the density adjustment factors from Section 4.1.2 and the transformed baseline density function from Eq. (4).

The DRN's density is assumed to follow a uniform distribution within each partitioned interval. Subsequently, we introduce the DRN density estimator as follows:

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\beta}) = \begin{cases} \sum_{k=1}^{K} \mathbb{1}_{\{y \in T_k\}} a_k(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\beta}) \cdot b_k(\boldsymbol{x}; \boldsymbol{\beta}) / |T_k|, & y \in [c_0, c_K), \\ f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}; \boldsymbol{\beta}), & \text{otherwise.} \end{cases} \tag{9}$$

A key feature of the DRN is its ability to maintain practical density estimation beyond the distributional refinement region—a notable advantage over DDR, whose distributional regression is limited to a fixed range. Specifically, for any y outside the range $[c_0, c_K)$, the DRN estimator reverts to the estimate of the baseline model: $f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\beta}) = f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}; \boldsymbol{\beta})$. As illustrated in Fig. 4, the DRN's refined density function mirrors the baseline model's density outside the refining region, highlighting the seamless baseline integration and transition. Building on the DRN density estimator as defined in Eq. (9), we derive the DRN cumulative distribution function (CDF) estimator:

$$F_{Y|\boldsymbol{X}}(y|\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\beta}) = \begin{cases} F_{Y|\boldsymbol{X}}(y|\boldsymbol{x}; \boldsymbol{\beta}), & y \in (-\infty, c_0), \\ F_{Y|\boldsymbol{X}}(c_0|\boldsymbol{x}; \boldsymbol{\beta}) + \int_{c_0}^{y} f_{Y|\boldsymbol{X}}(t|\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\beta})\, \mathrm{d}t, & y \in [c_0, c_K), \\ F_{Y|\boldsymbol{X}}(y|\boldsymbol{x}; \boldsymbol{\beta}), & y \in [c_K, \infty). \end{cases} \tag{10}$$

Specifically, the model directly adopts the baseline CDF for the response variable's values within $(-\infty, c_0)$ and beyond $[c_K, \infty)$. For values within the interval $[c_0, c_K)$, it incorporates adjustments based on the DRN's output to refine the CDF estimation, enhancing the model's accuracy and adaptability. A key insight is that actuarial modelling often requires a trade-off between accurately capturing the body versus the tail of a distribution. DRN overcomes this limitation by offering the best of both worlds. For example, one can start with a baseline model that captures the tail effectively and then refine the middle quantiles using the DRN.

4.2.2. Loss function and regularisation techniques

Selecting an appropriate loss function is critical for effectively training the DRN. We consider two potential choices for the loss function. One viable option is the Negative Log-Likelihood (NLL) loss function:

$$\mathcal{L}_{\text{NLL}}(\mathcal{D}, \boldsymbol{w}) = -\sum_{i=1}^{N} \ln\big(f_{Y|\boldsymbol{X}}(y_i|\boldsymbol{x}_i; \boldsymbol{w}, \boldsymbol{\beta})\big), \tag{11}$$

An alternative option is the Joint Binary Cross-Entropy (JBCE) loss function:

$$\begin{aligned}\mathcal{L}_{\text{JBCE}}(\mathcal{D}, \boldsymbol{w}) = -\sum_{k=1}^{K}\sum_{i=1}^{N}\Big[&\mathbb{1}_{\{y_i \le c_k\}} \ln F_{Y|\boldsymbol{X}}(c_k|\boldsymbol{x}_i; \boldsymbol{w}, \boldsymbol{\beta}) \\ &+ \Big(1 - \mathbb{1}_{\{y_i \le c_k\}}\Big) \ln\big(1 - F_{Y|\boldsymbol{X}}(c_k|\boldsymbol{x}_i; \boldsymbol{w}, \boldsymbol{\beta})\big)\Big]. \end{aligned} \tag{12}$$

The JBCE loss function emerges as the superior choice for training the DRN due to several compelling reasons. First, the JBCE loss ensures the model's performance remains stable across various partitioning configurations, meaning the distributional forecasting performance is less affected by the selection of cutpoints. This loss more effectively mitigates instability in 'zero-observation' intervals compared to the NLL loss (Li et al., 2021). Secondly, building on the previous reason, the JBCE loss allows for the allocation of additional cutpoints (Li et al., 2021), enhancing the density refinement capability of the neural network due to increased distributional flexibility. Finally, extending the classic distribution regression model (Foresi and Peracchi, 1995) to predict the entire conditional distribution requires the sequential training of binary classifiers at each cutpoint (Kneib et al., 2021). This methodology intuitively necessitates using the JBCE loss, prompting the DRN to minimise the cumulative binary classification losses across all partitioned intervals.

Building on the advantages of the JBCE loss function over NLL for training the DRN, we discuss the practical implementation of regularisation in this framework, driven by two primary motivations and desired properties. First, we create a controlled distributional continuum with respect to the baseline model. Second, we reduce the roughness of the density estimation (Good and Gaskins, 1971), enhancing the model's smoothness to ensure reliability and interpretability. Define $\hat{f}_{\text{Baseline}}$ and

$\hat{f}_{\text{DRN}}$ as the baseline and DRN density estimators, respectively, for an input instance $\boldsymbol{x}$. The regularised loss function is defined as follows:

$$\begin{aligned}\mathcal{L}^{*}_{\text{JBCE}}(\mathcal{D},\boldsymbol{w}) &= \mathcal{L}_{\text{JBCE}}(\mathcal{D},\boldsymbol{w})\\ &\quad+\alpha_1\cdot\underbrace{\frac{1}{N}\sum_{i=1}^{N} D_{\text{KL}}\big(\hat{f}_{\text{Baseline}}(y|\boldsymbol{x}_i)||\hat{f}_{\text{DRN}}(y|\boldsymbol{x}_i)\big)}_{\text{KL Penalty: Baseline Distributional Resemblance}} \qquad (13)\\ &\quad+\alpha_2\cdot\underbrace{\frac{1}{N}\sum_{i=1}^{N}\Phi_{c_0}^{c_K}\big(\hat{f}_{\text{DRN}}(y|\boldsymbol{x}_i)\big)}_{\text{Roughness Penalty: Density Smoothness}} \qquad (14)\\ &\quad+\alpha_3\cdot\underbrace{\frac{1}{N}\sum_{i=1}^{N}\big(\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x}_i;\boldsymbol{w},\boldsymbol{\beta}]-\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x}_i;\boldsymbol{\beta}]\big)^2}_{\text{Mean Penalty: Baseline Mean Resemblance}} \qquad (15)\end{aligned}$$

where $\alpha_1,\alpha_2,\alpha_3\geq 0$ are some penalty coefficients.

To rationalise the inclusion of these regularisation terms, we present both intuitive and theoretical justifications, complemented by empirical evidence from a study involving a simple synthetic dataset. The procedures for generating the synthetic dataset are detailed in Appendix C. The left panel of Fig. 5 compares the baseline model's density function, the non-regularised DRN's density, and the true density function for instance $X=(0.5,0.5)^{\top}$. This comparison reveals significant fluctuations in the non-regularised DRN density function across ordinates, indicating a risk of overfitting. Therefore, integrating a regularisation term that encourages the neural network to treat the baseline as "prior" knowledge can significantly decrease the likelihood of the model capturing noise as part of the learning process. More precisely, we apply a KL divergence penalty between the adjusted densities and baseline densities from (13):

$$\begin{aligned}&D_{\text{KL}}\big(\hat{f}_{\text{Baseline}}(y|\boldsymbol{x}_i)||\hat{f}_{\text{DRN}}(y|\boldsymbol{x}_i)\big)\\ &\quad=\int_{c_0}^{c_K}\hat{f}_{\text{Baseline}}(y|\boldsymbol{x}_i)\cdot\log\left(\frac{\hat{f}_{\text{Baseline}}(y|\boldsymbol{x}_i)}{\hat{f}_{\text{DRN}}(y|\boldsymbol{x}_i)}\right)\mathrm{d}y\\ &\quad=-\sum_{k=1}^{K}\int_{c_{k-1}}^{c_k}\hat{f}_{\text{Baseline}}(y|\boldsymbol{x}_i)\log(\hat{f}_{\text{DRN}}(y|\boldsymbol{x}_i))\,\mathrm{d}y+\mathcal{O}(1)\\ &\quad=-\sum_{k=1}^{K}\log(\hat{f}_{\text{DRN}}(c_{k-1}|\boldsymbol{x}_i))\int_{c_{k-1}}^{c_k}\hat{f}_{\text{Baseline}}(y|\boldsymbol{x}_i)\,\mathrm{d}y+\mathcal{O}(1) \qquad (16)\\ &\quad=-\sum_{k=1}^{K}\log(\hat{f}_{\text{DRN}}(c_{k-1}|\boldsymbol{x}_i))\\ &\qquad\underbrace{(\hat{F}_{\text{Baseline}}(c_k|\boldsymbol{x}_i)-\hat{F}_{\text{Baseline}}(c_{k-1}|\boldsymbol{x}_i))}_{\hat{b}_k(\boldsymbol{x}_i)\text{ in (5)}}+\mathcal{O}(1)\end{aligned}$$

This approach effectively endows the DRN with a well-structured baseline, mirroring a Bayesian approach that is particularly beneficial in scenarios characterised by sparse or limited data (Rossi and Allenby, 2003). The right panel of Fig. 5 illustrates how the adjusted density function considers the baseline density, aligning partially with the baseline estimation. As the regularisation coefficient α_1 increases towards infinity, the DRN's density estimation $\hat{f}_{\text{DRN}}(y|\boldsymbol{x})$ converges to the baseline model's estimation $\hat{f}_{\text{Baseline}}(y|\boldsymbol{x})$.

A notable challenge with highly flexible deep learning models is their propensity to generate fluctuating estimated density functions, as shown in Fig. 5. Ideally, a smoother density function is preferred for more effective interpretation and analysis, benefiting from the favourable statistical properties associated with either ordinary smooth or supersmooth distributions (Silverman, 1985; Fan, 1991). Thus, we aim to enhance the "overall smoothness" of the predicted density without neglecting the model's flexibility. The concept of "overall smoothness" here diverges from traditional definitions reliant on derivatives or characteristic functions. Instead, it qualitatively addresses two main issues: significant jumps between adjacent density values and "peaky" patterns. To mitigate these abrupt fluctuations, we introduce a roughness penalty inspired by smoothing splines (Craven and Wahba, 1978). This penalty is formulated as the square of the second-order difference in the density function, effectively reducing excessive and abrupt variations in the predicted densities:

$$\begin{aligned}\Phi_{c_0}^{c_K}(\hat{f}_{\text{DRN}}(y|\boldsymbol{x}_i)) &= \sum_{k=1}^{K-2}\big((\hat{f}_{\text{DRN}}(c_{k+1}|\boldsymbol{x}_i)-\hat{f}_{\text{DRN}}(c_k|\boldsymbol{x}_i))\\ &\quad-(\hat{f}_{\text{DRN}}(c_k|\boldsymbol{x}_i)-\hat{f}_{\text{DRN}}(c_{k-1}|\boldsymbol{x}_i))\big)^2. \qquad (17)\end{aligned}$$

The impact of the substantial roughness penalty is showcased in the left panel of Fig. 6. A large penalty value makes the estimated density overly "smooth" and less informative, diminishing the model's precision for distributional forecasting. Conversely, the right panel illustrates how a reasonable roughness penalty effectively mitigates excessive fluctuations and avoids unrealistic density jumps, thereby aligning the DRN's density estimate more closely with the actual density. This approach yields a more stable and interpretable density estimate, enhancing the model's ability to accurately capture the underlying distribution without overfitting to noise.

Despite the advantages of predicting the full conditional distribution, the mean remains a critical distributional property for actuarial applications, including pricing (e.g. pure premium) and reserving (e.g. central estimate). GLMs can provide unbiased estimators of the global population portfolio level mean, a feature known as the balance property. However, this desirable attribute may not be preserved when training neural networks (Wüthrich, 2022; Lindholm and Wüthrich, 2024). In certain cases, a GLM can adequately capture the mean at individual levels. Therefore, we integrate the component outlined in (15), which restrains the DRN from significant deviations from the mean predictions generated by the baseline model. This approach is identified as "mean regularisation". The implementation of mean regularisation can be practical when the baseline effectively models the mean.

Remark 1. The KL divergence expression in (16) is effectively a weighted sum of differences. In fact, two versions of the KL divergence exist, depending on the order with which the two distributions in (13) are taken, because the first will become the weighting function found in (16). When it is the distribution that is being fitted (in our case, the DRN density), one refers to "reverse KL", whereas when it is the benchmark distribution (in our case, the GLM density), one refers to "forward KL". The two formulations are not equivalent.

In traditional variational approximation (Corduneanu and Bishop, 2001), reverse KL divergence is commonly employed for computational simplicity (because the weight is often Gaussian, simplifying calculations). Our method adopts the forward KL divergence, as demonstrated in Eq. (13), effectively using a smooth baseline as the 'weight' to emphasise the DRN's density deviations from the baseline. Specifically, this weighting approach ensures logical consistency by prioritising alignment with the baseline forecasts, thus encouraging mean-seeking or mode-covering behaviours (Murphy, 2022). However, one could also employ reverse KL divergence in its approximate form, using the refined density as the weight. This approach is feasible when zero-forcing or mode-seeking is ideal, i.e., when deviation from the baseline is desirable.

Remark 2. In DRN, overfitting risk is controlled in two main ways. First, the training loss includes distributional regularisation terms that anchor the refinement to the baseline: a forward KL penalty that penalises deviations from the baseline density (13), a roughness penalty that discourages spiky refinements (14), and an optional mean penalty that preserves the baseline mean when appropriate (15). Second, we use standard generalisation devices such as dropout and early stopping. Our held-out test results indicate that these mechanisms are effective at mitigating overfitting. For the synthetic dataset in Section 5, DRN improves strictly proper scores relative to DDR and is competitive with MDN (a similarly complex and competent neural network for distributional regression); see Table 1 and also Remark 3. For the real dataset

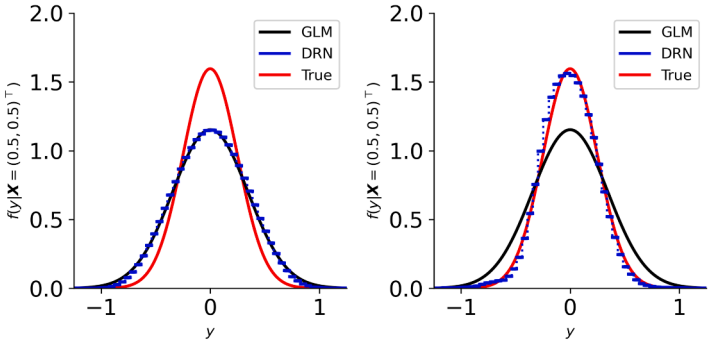


Fig. 6. The comparison showcases DRN density functions under different roughness penalty intensities for the instance $X = (0.5, 0.5)^\top$. The left graph illustrates the outcome with extremely high regularisation ($\alpha_2 = 1.0$), leading to a highly smooth but potentially less informative density estimate. The right graph, on the other hand, presents the impact of moderate regularisation ($\alpha_2 = 0.0005$), which offers a balanced density estimate that aligns more closely with the true distribution. In both graphs, the DRN density function (blue) is compared against the baseline (black) and true density (red) functions. The KL divergence penalty coefficient is fixed at 0.001. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

in Section 6, across 100 random splits, DRN attains a better average rank on almost all metrics (Fig. 11) than DDR, again indicating good out-of-sample performance. Moreover, the baseline-dependency study in Appendix D shows DRN's robustness in small to medium samples: with limited training data, DRN can consistently outperform DDR and MDN when provided a reasonable baseline. These results may suggest improved generalisation properties for small to medium sample sizes.

4.2.3. Model evaluation

For assessing the DRN's distributional forecasting performance, we propose using the Continuous Ranked Probability Score (CRPS) (Gneiting and Raftery, 2007), Negative Log-Likelihood (NLL), and Quantile Loss (QL). The Root mean squared error (RMSE) is utilised to assess the accuracy of the central estimate. The detailed formulas for these evaluation metrics are presented in Appendix G. Note that both the CRPS and NLL or logarithmic score, are strictly proper scoring rules that encourage the predicted distribution to be equal to the actual underlying distribution so forecasts can be precise and optimal.

In addition to the numerical evaluation metrics, visual plots such as quantile residuals and calibration plots can be employed to assess model performance. The quantile residuals plot (Dunn and Smyth, 1996) offers a diagnostic tool for evaluating the adequacy of fitting regression models. Actuarial modelling widely uses it to measure a model's sufficiency in fitting. To construct a quantile residuals plot, we first determine the CDF values of the observed responses, i.e., $\hat{F}(y_i|\boldsymbol{x}_i)$'s. Then we use the inverse of the standard normal distribution to obtain the quantile residuals, i.e., compute $r_i = \Phi^{-1}(\hat{F}(y_i|\boldsymbol{x}_i))$'s. Finally, we plot the obtained quantile residuals on a QQ-plot. A well-fitted model's quantile residuals will closely align with the 45-degree line on a QQ-plot.

Further, neural networks could be poorly calibrated: the p% prediction interval might not contain the actual observation p% of the time. A well-calibrated model is defined in Definition 1, elaborated in Appendix I. A calibration plot serves as a valuable tool for evaluating the calibration of a probabilistic model. This plot contrasts the empirical cumulative probabilities with the predicted ones for all observations. Ideally, a well-calibrated model should have the points closely aligned to the 45-degree line. Deviations from this line can indicate an overestimation or underestimation of the predicted probabilities. Strategies may include model refinement or applying recalibration techniques to enhance calibration, as Romano et al. (2019) and Kuleshov et al. (2018) discussed.

4.3. Implementation example

The mathematical framework presented in this section can be implemented using the `drn` Python package (Dong and Laub, 2024). The following code snippet demonstrates how the equations translate into practical implementation:

```
from drn import GLM, DRN
# Train baseline GLM model (Section 4.1.1)
glm_model = GLM("gamma").fit(X_train, y_train)
# Create and train DRN using the baseline (Section 4.1.2)
drn_model = DRN(glm_model).fit(X_train, y_train)
# Generate distributional predictions (Section 4.2)
predictions = drn_model.predict(X_test)
mean_pred = predictions.mean
quantiles = predictions.quantiles([10, 50, 90])
```

This implementation directly corresponds to the DRN architecture in Fig. 3, where the baseline GLM provides the initial distributional forecast that is subsequently refined by the neural network component.

5. Numerical study using synthetic data

This section demonstrates the DRN's capacity for distributional forecasting using a synthetic dataset. The process for data generation is detailed in Section 5.1. Model specifications for competing distributional regression models, along with their hyperparameter tuning procedures, are detailed in Section 5.2. The DRN's performance regarding distributional flexibility and interpretability is comprehensively examined in Section 5.3.

5.1. Data generation

This section describes the generation of a synthetic regression dataset, $\mathcal{D} = (\boldsymbol{x}_i, y_i)_{i=1}^N$, representing the i.i.d. random vector $(\boldsymbol{X}, Y)$. In this dataset, the feature vectors follow a multivariate Gaussian distribution, whereas the response variable combines gamma and lognormal distributions:

$$\begin{aligned}\boldsymbol{X} &= (X_1, X_2)^\top \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),\\ Y &= \text{Gamma}(\mu(\boldsymbol{X}), \phi(\boldsymbol{X})) + \text{LogNormal}(\log(\mu(\boldsymbol{X})), \phi(\boldsymbol{X})),\end{aligned} \tag{18}$$

where,

- the features exhibit correlation, defined by parameters:

$$\boldsymbol{\mu} = (0,0)^\top, \boldsymbol{\Sigma} = \begin{bmatrix} 0.25^2 & 0.25^3 \\ 0.25^3 & 0.25^2 \end{bmatrix}, \tag{19}$$

  i.e., $\mu_1 = \mu_2 = 0, \sigma_1 = 0.25, \sigma_2 = 0.25, \rho = 0.25$, and
- the dataset specifies mean and dispersion functions for $Y|\boldsymbol{X}$ as follows:

$$\mu(\boldsymbol{X}) = \exp(-X_1 + X_2) \tag{20}$$

$$\phi(\boldsymbol{X}) = \exp(X_1)/(1 + \exp(X_1 \cdot X_2)). \tag{21}$$

Drawing parallels to practical scenarios, such as motor insurance, gamma and lognormal random variables can be interpreted as administrative costs and actual incurred losses, respectively. In this context, the feature X_1 can be conceptualised as the standardised age of the policyholder's license, while X_2 refers to the standardised value of the policyholder's vehicle. Basing loss predictions exclusively on these two risk factors is insufficient. Nonetheless, this simplification is deliberate, aiming to highlight two distinct behaviours: (i) the DRN makes reasonable adjustments to the mean predictions, mostly accurately but not perfectly forecasted by the baseline GLM; (ii) the DRN enhances the accuracy of predictions at various quantile levels, surpassing the GLM's performance, which falls short in modelling varying dispersion.

The dataset consists of 12,000 randomly sampled training observations and 4000 for validation. Since the data-generating process is known, we generate 50,000 test observations for evaluation.

5.2. Model specification and training

This section details the specifications and training procedures, including hyperparameter tuning, for the five models under comparison: GLM, CANN, MDN, DDR, and DRN.

The GLM is the standard in practice for actuarial regression problems. For training the DRN within the proposed DRN framework, we select a gamma GLM with a log link function as the baseline. For all deep-learning models, we employ the `Adam` optimiser (Kingma and Ba, 2014). Early stopping is set with a tolerance of 30 epochs. We employ the Leaky Rectified Linear Unit (`LeakyReLU`) as the activation function between the hidden layers, as explained in Appendix E. The DDR's output layer utilises a `softmax` activation function to establish probabilities for each partitioned interval. Both CANN and our proposed DRN utilise the gamma GLM as the baseline. We pick the `exponential` activation function for the CANN's output layer to output the factors adjusting the means estimated by the gamma GLM. The DRN's output layer uses a `Linear` activation function. For the MDN's output layer, we implement a `softmax` activation function to determine the mixing weights and two `softplus` activation functions to find each component's shape and scale parameters.

Deep architectures can exhibit vanishing gradients (Glorot and Bengio, 2010). In our setting, gradients for DRN parameters arise from the data-fit loss (NLL or JBCE) and distributional regularisation terms. The baseline enters the objective as a fixed reference; hence, baseline complexity does not create additional back-propagation paths. Empirically, we did not encounter vanishing-gradient issues with LeakyReLU and `Adam`. If this were to arise, a standard remedy is to introduce an explicit skip connection technique to preserve gradient flow and ease optimisation (Srivastava et al., 2015; He et al., 2016).

We employ the `skopt.gp_minimize` function from `Scikit-Optimize` (`skopt`) for tuning hyperparameters. This method employs Gaussian process-based Bayesian optimisation, which efficiently navigates the trade-offs between exploration of the hyperparameter space and exploitation of known good regions. Table F.8 in Appendix F presents a detailed comparison of the hyperparameter ranges employed across various advanced models. The objective function for hyperparameter tuning is the average Continuous Ranked Probability Score (CRPS) loss. Each model is subjected to a total of 125 evaluations and 25 random initialisations. Further details are provided in Appendix F. The final hyperparameter settings and tuning durations for all competing models are summarised in Table F.9 in Appendix F.

5.3. Distributional flexibility and interpretability

This section explores the DRN framework's distributional flexibility and interpretability, two major aspects we are concerned about in distributional regression. Section 5.3.1 displays its exceptional distributional flexibility by leveraging various evaluation metrics. Section 5.3.2 illustrates model-specific plots that support trust and interpretability, including visual comparisons between the baseline and refined distributions and decompositions of key distributional properties.

5.3.1. Flexibility: model performance

To assess both probabilistic and point forecasts, we employ four evaluation metrics: Continuous Ranked Probability Score (CRPS), Negative Log-Likelihood (NLL), Root Mean Squared Error (RMSE), and 90% Quantile Loss (90% QL). Table 1 below demonstrates the results of different models.

Remark 3. Under the data-generating process in Eq. (18), where $Y|\boldsymbol{X}$ is a sum of Gamma and lognormal components with relatively small mean and dispersion, the Gamma MDN is the most flexible model in our comparison and can be viewed as a natural "gold standard" for distributional performance in this context, at the cost of interpretability (relative to GLM, CANN, and DRN). In this light, it is encouraging that the DRN attains NLL and CRPS that are very close to those of the MDN, with no statistically significant differences according to the Wilcoxon tests, as shown in Table 1. DRN also clearly dominates CANN and DDR across all distributional metrics. Interestingly, DRN does not attain the lowest metrics on the validation set–even though it clearly outperforms DDR and is competitive with the Gamma MDN on the test set. This pattern is consistent with the explicit distributional regularisation in DRN , which slightly restricts flexibility and favours models that generalise well (see Remark 2), whereas the more weakly regularised MDN and DDR can fit a particular validation split more closely at the risk of mild overfitting. Moreover, in more challenging settings where the response distribution deviates more strongly from the Gamma family (Section 5.4), DRNs can even outperform Gamma MDNs in CRPS, RMSE and QL90 for small and moderate training sizes, while remaining only slightly inferior in NLL.

To investigate hyperparameter sensitivity, we selected the top four DRN configurations based on validation CRPS (see Table F.10) and compared their predictive performance on both validation and test sets (Table 2). Despite noticeable differences in the chosen hyperparameters, the resulting models yield highly consistent predictive performance, particularly on the test set, demonstrating robustness to hyperparameter variation and suggesting that DRN's performance is not overly reliant on fine-tuned settings.

The left panel of Fig. 7 presents the calibration plots. The DRN_1's points are closely aligned to the 45-degree line, and the calibration score is the lowest across all models. The right panel displays the quantile residual plots. The points are relatively aligned to the 45-degree line for the DRN.

5.3.2. Interpretability: trust and transparency

The scope of interpretability comprises two primary categories: local and global; see (Molnar, 2020; Burkart and Huber, 2021). Local interpretability concentrates on clarifying the rationale behind specific model predictions, enabling detailed analysis of particular decisions. Meanwhile, global interpretability aims to reveal the model's holistic logic, offering insights into its behaviour across a wide range of scenarios. The GLM and GAMLSS are regarded as interpretable for two major reasons: i) the modelling "simplicity" is guaranteed due to the small number of distributional parameters required to define the distribution; ii) distributional properties are relatively straightforward transformations of the features, facilitating a clear understanding of feature contributions to predictions of mean, variance, etc. While DRN does not claim to satisfy these two characteristics in the same way as GLM or GAMLSS, it can be well-trusted because of its inherently "refining" characteristic, whereby the predicted distributions, hence the key distributional properties, remain closely anchored to those predicted by the baseline models. We will demonstrate this locally and globally.

The DRN framework enables users to visualise instance-specific deviations from the baseline distribution, enabling transparent interpretation of local distributional refinements. Such a functionality effectively mitigates the concern of the DRN's reliance on a significantly larger

Table 1
Model comparisons based on various evaluation metrics. To robustly assess if the DRN significantly enhances distributional forecasting compared to the baseline GLM, the Wilcoxon Signed-Rank Test, introduced by Wilcoxon (1945), is applied to NLL, CRPS, squared errors and 90% QL scores. The bolded metrics represent the best-performing model for each respective column. This non-parametric method evaluates the statistical significance of median differences between the performance metrics of the two models. The statement 'Model1 < Model2' implies testing the null hypothesis that the median metrics of both models are equal against the alternative that Model1's median is lower. A p-value less than 0.05 is indicated with one star (*), less than 0.01 with two stars (**), and less than 0.001 with three stars (***). Note that the table entries report mean scores over the validation and test sets, whereas the Wilcoxon signed-rank tests are applied to the paired observation-wise differences in scores and, for hypotheses of the form "DRN < MDN", effectively test whether the median of (DRN score - MDN score) is negative. Consequently, small discrepancies between the sign of the mean difference and the Wilcoxon outcome may arise when the distribution of these differences is skewed. For a given model, if any of the model's predicted distributions assign zero density to an observed y_i value in an evaluation dataset, then that model's NLL will be infinite for that dataset.

	$\mathcal{D}_{\text{Validation}}$				$\mathcal{D}_{\text{Test}}$			
Model \ Metrics	NLL	CRPS	RMSE	90% QL	NLL	CRPS	RMSE	90% QL
GLM	1.2896	0.5254	0.9872	0.2046	1.2912	0.5282	**1.0033**	0.2085
CANN	1.2878	0.5249	0.9868	0.2050	1.2902	0.5282	1.0033	0.2092
MDN	**1.2535**	**0.5207**	0.9877	**0.2018**	**1.2530**	**0.5238**	1.0042	**0.2057**
DDR	∞	0.5208	**0.9870**	0.2018	∞	0.5245	1.0042	0.2063
DRN (DRN_1 in Table F.10)	1.2643	0.5211	0.9874	0.2020	1.2631	0.5241	1.0036	0.2061
	$\mathcal{D}_{\text{Validation}}$				$\mathcal{D}_{\text{Test}}$			
Hypothesis \ Metrics	NLL	CRPS	RMSE	90% QL	NLL	CRPS	RMSE	90% QL
DRN < GLM	(***)	(***)	()	(***)	(***)	(***)	()	(***)
DRN < CANN	(***)	(***)	(***)	(***)	(***)	(***)	(***)	(***)
DRN < MDN	()	()	(***)	(***)	()	()	(***)	(***)
DRN < DDR	(***)	(***)	(***)	(***)	(***)	(***)	(***)	(***)

Table 2
Model comparisons based on DRN models trained using the hyperparameter settings listed in Table F.10.

	$\mathcal{D}_{\text{Validation}}$				$\mathcal{D}_{\text{Test}}$			
Model \ Metrics	NLL	CRPS	RMSE	90% QL	NLL	CRPS	RMSE	90% QL
DRN_1	1.2643	0.5211	0.9874	0.2020	1.2631	0.5241	1.0036	0.2061
DRN_2	1.2615	0.5213	0.9880	0.2019	1.2614	0.5241	1.0044	0.2061
DRN_3	1.2666	0.5213	0.9874	0.2019	1.2626	0.5241	1.0037	0.2059
DRN_4	1.2634	0.5213	0.9877	0.2022	1.2655	0.5241	1.0041	0.2062

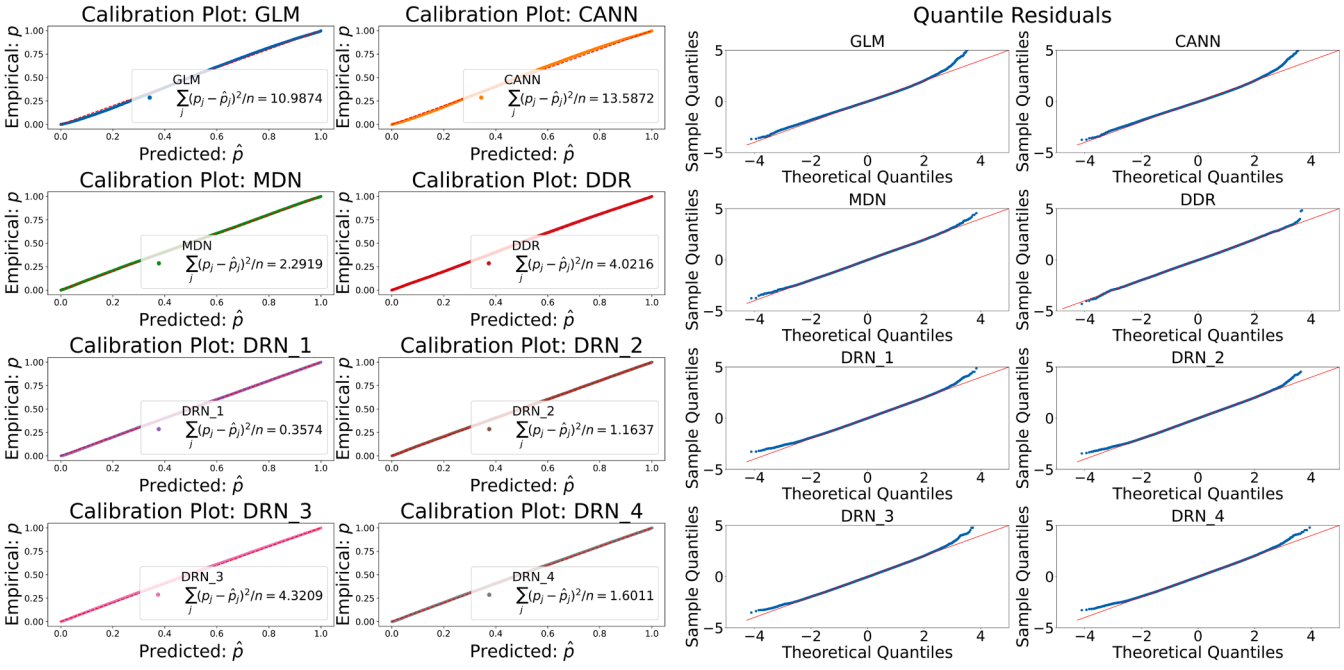


Fig. 7. Calibration (left) and quantile residuals (right) plots generated on the test dataset.

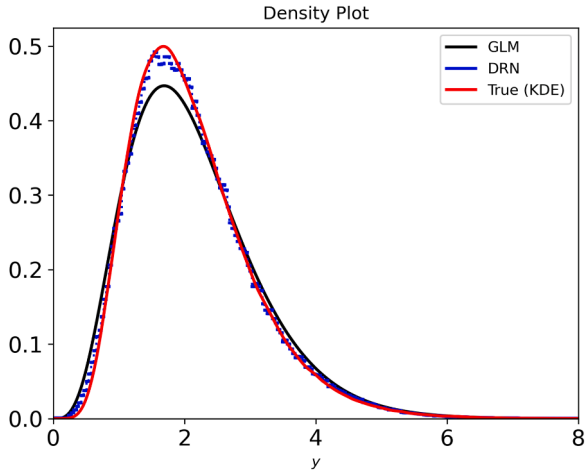


Fig. 8. Comparison of predicted conditional densities of $Y|\boldsymbol{x}^*$ by the baseline model (black) and the DRN (blue) with the true density (red), where $\boldsymbol{x}^* = (0.0, 0.0)^\top$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

set of distributional parameters. Through this visualisation, users gain clarity on the model's predictions despite the complexity introduced by the increased parameter count. For illustration, we examine an instance $\boldsymbol{x}^* = (x_1^*, x_2^*)^\top = (0.0, 0.0)^\top$, characterised by feature values fixed at their means. Given our control over the synthetic dataset generation, we can directly compare the true distribution with those generated by the baseline and DRN models. Fig. 8 showcases this comparison: the true distribution is represented in red, the baseline in black, and the DRN's prediction in blue. Qualitatively, the DRN's distribution aligns more closely with the true distribution than the baseline's. Such a visualisation allows us to trust the model's distributional flexibility.

We can also employ Kernel SHAP (Lundberg and Lee, 2017) as a supplementary analytical tool (detailed in Appendix H). This helps us assess whether and how DRN diverges from the baseline model in terms of predicting key distributional properties. The detailed Kernel SHAP analysis is provided in Appendix H.1. Although Kernel SHAP offers substantial computational efficiency by approximating Shapley values using marginal distributions, and provides useful insights through interpretable decompositions of predictions, it may not fully capture complex non-linear interactions. Its reliance on the feature independence assumption can be inappropriate in practice. In our context, Kernel SHAP approximates the value functions using the marginal distributions of X_1 and X_2 rather than their conditional distributions $X_1|X_2$ and $X_2|X_1$. Consequently, the interaction between these features is not fully captured, potentially leading to deviations from the expected relationships. Hence, one should interpret and leverage the created plots carefully and objectively. Furthermore, even when computed exactly, Shapley values can misrepresent true feature importance by assigning non-zero scores to irrelevant features or underestimating the contributions of relevant ones (Marques-Silva and Huang, 2024). In a way, the trade-off between computational efficiency and accuracy in current Shapley-based methods highlights the value of DRN's baseline integration–its controlled refinement, guided by principled regularisation, ensures both trustworthiness in distributional prediction.

5.4. Baseline dependency and practical guidelines

In many practical actuarial applications, a GLM remains a trusted standard approach because it is well understood and highly interpretable. In those instances, a baseline is readily available. It is then sufficient to examine reasonable baseline distributional assumptions and, where feasible, apply feature engineering to identify a sensible starting point for DRN refinements. That being said, the baseline-dependency study in Appendix D suggests that the DRN is generally robust to the choice of baseline.

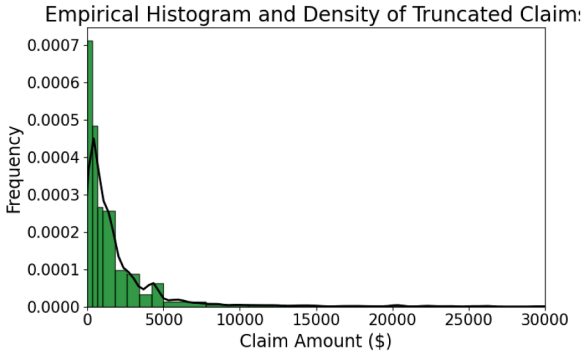


Fig. 9. The empirical histogram and density of claim amounts censored at $30,000.

6. Illustrating DRN's practical application with real data

To showcase the practical applicability of our proposed DRN framework in an insurance context, we evaluate its performance in modelling claim severity. For this purpose, we utilise an actuarial dataset with the response variable as claim amounts (positive random variables) and high-dimensional features consisting of both categorical and numerical features. Specifically, we employ the `freMPL1` dataset from the `R` package `CASdatasets` (Dutang and Charpentier, 2025), focusing on modelling claim amounts given the features.

We selected this particular dataset for three main reasons. First, the empirical conditional density of claim amounts, given the features, exhibits characteristics such as multimodality and heavy-tailedness. These features represent the frequent challenges in actuarial modelling. Second, the dataset possesses a moderate number of observations and a mix of numerical and categorical features. It comprises 2205 observations, three continuous features, four binary features, and eleven categorical features. Lastly, this dataset was employed by Delong et al. (2021) to investigate the Gamma MDN, one of our benchmark models.

Section 6.1 delves into Data Preprocessing and Exploratory Data Analysis. In Section 6.2, we detail the procedure for model training, leading into Section 6.3, which examines the performance of these models.

6.1. Data preprocessing and exploratory data analysis

Fig. 9 presents the empirical histogram and density of claim amounts censored at a maximum value of $30,000. It's important to note that the applied truncation and density smoothing technique somewhat conceal the distribution's notable characteristics, such as heavy-tailedness and multimodality.

Comprehensive data cleaning and transformation steps are employed. First, we scale the response variable, `ClaimAmount`, which is reduced by a factor of 1,000. Additionally, we discard several columns, namely `RecordBeg`, `RecordEnd`, `ClaimInd`, and `Garage`, with the exclusion of `Garage` due to many missing values. Regarding the feature `VehAge` (vehicle age), which was originally provided in categories, it is mapped to specific numerical values. Notably, '6-7' is assigned with a value of 6 and '10+' is assigned with a value of 11. We convert categorical speed ranges into a numeric sequence for the `VehMaxSpeed` (maximum speed range of the vehicle) feature. For example, '1-130 km/h' is converted to an integer value of 1, '130-140 km/h' has an integer value of 2, and so on.

For feature preprocessing, we apply one-hot encoding to categorical variables such as `HasKmLimit`, `Gender`, `SocioCateg` and `MariStat`, among others. The dataset is subsequently divided into training, vali-

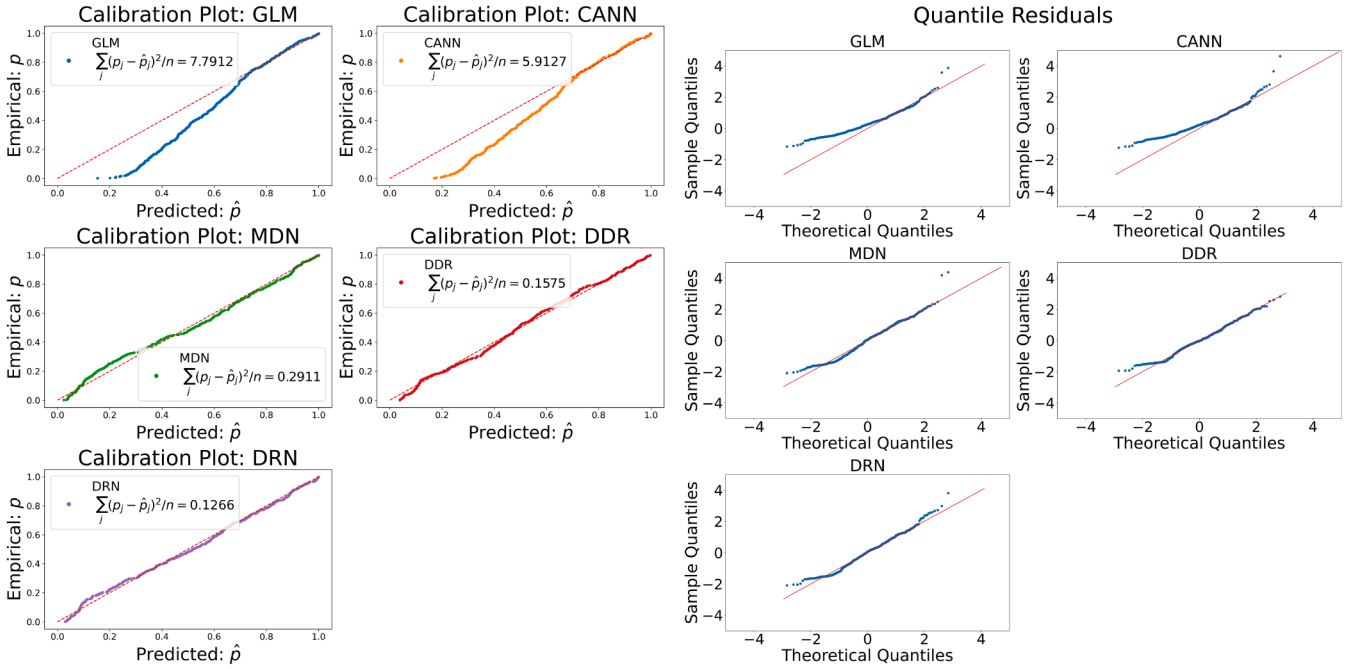


Fig. 10. Calibration (left) and quantile residuals (right) plots generated on the test dataset.

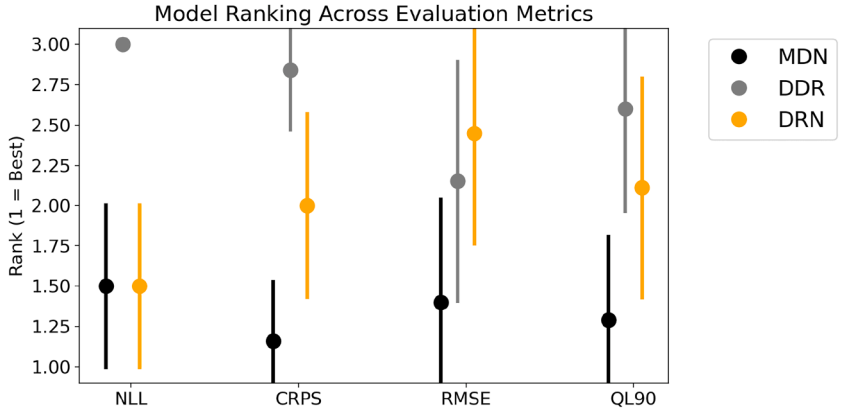


Fig. 11. Mean rank and one standard deviation of each model across NLL, CRPS, RMSE, and QL90, evaluated over 100 random data splits. Dots indicate average ranks on the test sets; vertical lines represent variability (± standard deviation) across splits. Lower positions indicate better performance.

dation, and test sets, with 60%, 20%, and 20% for training, validation, and testing, respectively.

6.2. *Model training and hyperparameter tuning*

We select a gamma GLM with a log link function as the baseline for the DRN. The early stopping patience parameter is fixed at 50 for all neural networks. We employ the first-order `Adam` optimiser. The activation functions for the hidden and output layers match the ones highlighted in Section 5.2. The comparison of the hyperparameter ranges used across various advanced models is presented in Table F.11 within Appendix F. Each model undergoes a total of 125 evaluations with 25 random initialisations; see Appendix F for details about what "evaluations" and "initialisations" are within the Gaussian Process-based hyperparameter tuning procedure. The tuned hyperparameters are summarised in Table F.12 in Appendix F.

6.3. *Distributional flexibility*

Table 3 reports single-split results for the claim-severity dataset. On this particular split, the fully flexible MDN and DDR attain the lowest test NLL and CRPS. This is unsurprising: both models are unconstrained black-box neural networks that are free to optimise distributional flexibility without being anchored to a baseline. By contrast, the DRN is explicitly regularised towards the Gamma-GLM baseline, so a small loss in raw flexibility is expected. Even so, DRN delivers substantial gains over the actuarial GLM and CANN benchmarks across all distributional metrics while remaining competitive to the best MDN/DDR scores. This pattern is consistent with the modelling goals of the DRN: our aim is not necessarily to dominate fully black-box models such as MDN and DDR, but to refine a trusted, transparent baseline while retaining most of the achievable predictive performance.

Fig. 10 displays the calibration and quantile residual plots. The DRN model's calibration points, shown in the left panel, are close to the 45-degree line on average, and the calibration score for the DRN is the lowest. This indicates that the DRN provides the best calibration. Similarly, the quantile residual plots in the right panel confirm that the DRN model aligns reasonably well with the 45-degree line.

However, we note that results from Table 3 and Fig. 10 should be interpreted with caution. Due to the inherent variability from data splitting and the limited sizes of training and test sets, there can be substantial fluctuations in performance metrics, which means the table and fig-

Table 3
Model comparisons based on various evaluation metrics. The bolded metrics represent the best-performing model for each respective column. The Wilcoxon Signed-Rank Test (Wilcoxon, 1945) is applied to NLL, CRPS, squared errors and 90% QL scores. The statement 'Model1 < Model2' implies testing the null hypothesis that the median metrics of both models are equal against the alternative that Model1's median is lower. A p-value less than 0.05 is indicated with one star (*), less than 0.01 with two stars (**), and less than 0.001 with three stars (***).

	$\mathcal{D}_{\text{Validation}}$				$\mathcal{D}_{\text{Test}}$			
Model \ Metrics	NLL	CRPS	RMSE	90% QL	NLL	CRPS	RMSE	90% QL
GLM	1.9302	1.6466	5.2262	1.0626	1.9509	1.5059	4.3206	0.9060
CANN	1.8780	1.6157	5.1399	1.0378	1.9088	1.4771	4.1682	0.9123
MDN	**1.6568**	**1.5555**	**5.0568**	**0.9935**	1.6820	1.4005	4.0392	**0.8568**
DDR	1.6677	1.5601	5.0726	1.0053	**1.6665**	**1.3946**	**4.0193**	0.8756
DRN	1.6931	1.5765	5.1628	1.0235	1.7107	1.4101	4.1856	0.8623
	$\mathcal{D}_{\text{Validation}}$				$\mathcal{D}_{\text{Test}}$			
Hypothesis \ Metrics	NLL	CRPS	RMSE	90% QL	NLL	CRPS	RMSE	90% QL
DRN < GLM	(***)	(***)	()	(***)	(***)	(***)	(*)	(***)
DRN < CANN	(***)	(***)	()	(***)	(***)	(***)	()	(***)
DRN < MDN	()	()	()	()	()	()	()	()
DRN < DDR	()	()	()	()	()	()	()	()

ure provide an approximate indication of the models' true generalisation capability.

Hence, we additionally conduct repeated experiments across 100 random data splits. For each split, we train all models using the following hyperparameter settings. All neural networks use 3 hidden layers with 256 units per layer, a dropout rate of 0.2, a learning rate of 1×10^{-3}, and a batch size of 128. Early stopping is applied with a patience of 50 epochs. For DRN and DDR models, the cutpoint proportion is set to 0.4 to control output granularity. For DRN, the KL, roughness, and mean penalties are fixed at 5×10^{-3}. The minimum number of observations per interval is set to 5. We evaluate their performance on the corresponding test set. For every evaluation metric (NLL, CRPS, RMSE, and QL90), we assign ranks to the models based on their relative performance within each split. Fig. 11 summarises the results by displaying the mean rank and one standard deviation for each model across all splits. DRN consistently outperforms DDR in average rank across NLL, RMSE, and QL90, falling behind only the fully flexible MDN.

For a complementary analysis of DRN's local and global distributional interpretability on this dataset using Kernel SHAP, see Appendix H.2.

7. Conclusions

In this paper, we introduce the Distributional Refinement Network (DRN), a novel distributional regression model that integrates inherently interpretable parametric models with flexible semiparametric deep learning techniques. The DRN framework demonstrates potential in optimising distributional flexibility while preserving a level of interpretability, effectively navigating a main challenge in distributional regression. The DRN refines distributional forecasts from a baseline regression model (a GLM in this paper) to accurately capture the varying impacts of features across all quantiles. We detail our methodological approach for training the DRN, covering the specialised partitioning algorithm, the selection rationale for the loss function, and the integration of regularisation techniques.

This paper's primary motivation was the need for improved distributional forecasting methods relative to traditional GLM-based models. Our method can be seen as an improvement on Deep Distribution Regression (DDR). Specifically, the DRN offers three significant improvements over DDR through baseline model integration: (i) It optimises the capability to accurately forecast all crucial distributional properties by integrating a reasonable and interpretable baseline model, allowing for localised refinements while ensuring well-defined distributions beyond the refining region. This overcomes DDR's limitations in both learning the distribution from scratch and analysing tail behaviour. (ii) It employs regularisation techniques, such as KL divergence, to create a controlled continuum of anchoring to the baseline model. This effectively mitigates overfitting and enhances robustness beyond what DDR can offer; and (iii) It enables transparent comparison between the baseline and the refined distributions, offering interpretable insights and meaningful diagnostics not possible with DDR's fully black-box approach.

CRediT authorship contribution statement

Benjamin Avanzi: Writing – review & editing, Visualization, Validation, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization; **Eric T. Dong:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Conceptualization; **Patrick J. Laub:** Writing – review & editing, Visualization, Validation, Software, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization; **Bernard Wong:** Writing – review & editing, Visualization, Validation, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Generative AI

During the preparation of the work the authors used Grammarly for grammar checking and ChatGPT to identify potential errors in Python code. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Data and Code

The synthetic datasets were generated in Python. The real dataset `freMPL1` was obtained from the `R` package `CASdatasets`. Results presented in this paper can be replicated using the Python code available at https://github.com/agi-lab/drn.

Python package `drn`

The GitHub repository can be found at https://github.com/EricTianDong/drn. This package includes different distributional regression models, such as GLM, CANN, MDN, and DDR, as well as the proposed DRN model from this paper. A Python package named `drn` is available on PIP.

Data Availability

I have shared the link to my data and code in the paper.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

An earlier version of this paper was presented at the 2023 Australasian Actuarial Education and Research Symposium in Wellington, New Zealand. The authors are grateful for constructive comments received from colleagues who attended those events. We also thank two anonymous referees, whose insightful comments led to significant improvements of the paper. Avanzi and Wong acknowledge financial support under Australian Research Council's Discovery Project DP200101859 funding scheme. Dong acknowledges financial support by the Australian Government Research Training program, as well as the UNSW Business School through supplementary scholarships. The views expressed herein are those of the authors and are not necessarily those of the supporting organisations.

Appendix A. Distributional Regression Models

A.1. Generalised linear model

The GLM (Nelder and Wedderburn, 1972) stands as one of the most classical distributional regression models. A GLM employs a regression function, $\mu_{\boldsymbol{\beta}} : \mathbb{R}^p \to \mathbb{R}$, to model the mean of the response variable, assuming it follows an exponential family distribution:

$$\mu_{\boldsymbol{\beta}}(\boldsymbol{x}) = g^{-1}\left(\beta_0 + \sum_{j=1}^{p} \beta_j x_j\right). \tag{A.1}$$

Here, $\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_p)^\top \in \mathbb{R}^{p+1}$ denotes the regression coefficient vector and $g^{-1}(\cdot)$ is the inverse link function, which maps the linear combination of the features $\boldsymbol{x}$ to the mean of the response variable.

A.2. Generalised additive models for location, scale, and shape

GAMLSS (Rigby and Stasinopoulos, 2005) is a general distributional regression framework. It enhances distributional flexibility beyond traditional GLMs by supporting more flexible distributional assumptions that extend well beyond the standard exponential family distributions. GAMLSS assumes that given a feature set $\boldsymbol{X} = \boldsymbol{x}$, the response variable Y follows a distribution $\mathcal{F}$ characterised by parameters $(\theta_1(\boldsymbol{x}), \dots, \theta_K(\boldsymbol{x}))$. Here, GAMLSS constructs K separate regression functions, each modelling a different parameter of the underlying distribution. The kth regression function $\mathcal{M}_{\boldsymbol{w}_{\text{GAMLSS},k}} : \mathbb{R}^p \to \mathbb{R}$ models the kth distributional parameter:

$$\mathcal{M}_{\boldsymbol{w}_{\text{GAMLSS},k}}(\boldsymbol{x}) = \theta_k(\boldsymbol{x}; \boldsymbol{\beta}_k, \boldsymbol{\gamma}_k) = g_k^{-1}\left(\beta_{k,0} + \sum_{j=1}^{p} \beta_{k,j} x_j + \sum_{j=1}^{J_k} h_{k,j}(\boldsymbol{x}; \boldsymbol{\gamma}_{k,j})\right), \tag{A.2}$$

where g_k^{-1} represents the kth inverse link function, $\boldsymbol{\beta}_k = (\beta_{k,0}, \beta_{k,1}, \dots, \beta_{k,p})^\top \in \mathbb{R}^{p+1}$ denotes the linear vector of coefficients for the kth parameter, and $\boldsymbol{\gamma}_{k,j}$ represents the regression coefficient vector for the jth smooth non-parametric function predicting the kth parameter, i.e., $h_{k,j}$.

A.3. Distributional neural network

A simple feedforward neural network $\mathcal{M}_{\boldsymbol{w}} : \mathbb{R}^p \to \mathbb{R}^K$ follows

$$\mathcal{M}_{\boldsymbol{w}}(\boldsymbol{x}) = a^{(L,L+1)} \circ \cdots \circ a^{(1,2)} \circ a^{(0,1)}(\boldsymbol{x}), \tag{A.3}$$

where L represents the number of hidden layers, and K is the number of neurons in the output layer. For $l \in \{0, \dots, L\}$, we consider the layer mapping $a^{(l,l+1)} : \mathbb{R}^{q^{(l)}} \to \mathbb{R}^{q^{(l+1)}}$ such that

$$a^{(l,l+1)}(\boldsymbol{z}^{(l)}) = \phi^{(l+1)}\left(\left(b_1^{(l+1)} + \langle \boldsymbol{w}_1^{(l,l+1)}, \boldsymbol{z}^{(l)} \rangle, \dots, b_{q^{(l+1)}}^{(l+1)} + \langle \boldsymbol{w}_{q^{(l+1)}}^{(l,l+1)}, \boldsymbol{z}^{(l)} \rangle\right)^\top\right), \tag{A.4}$$

where $q^{(l)}$ is the number of neurons and $b_k^{(l)}$ is the bias term for the kth neuron in the lth layer for $k \in \{1, \dots, q^{(l)}\}$. The activation function for the $(l+1)$th layer is denoted as $\phi^{(l+1)} : \mathbb{R}^{q^{(l+1)}} \to \mathbb{R}^{q^{(l+1)}}$. $\boldsymbol{z}^{(l)} \in \mathbb{R}^{q^{(l)}}$ denotes the neuron values for the lth layer, where $\boldsymbol{z}^{(0)} = \boldsymbol{x}$ and $\boldsymbol{z}^{(L+1)} = \mathcal{M}_{\boldsymbol{w}}(\boldsymbol{x})$. Further, we define the weights connecting the neurons in the lth layer to the kth neuron in the $l+1$th layer:

$$\boldsymbol{w}_k^{(l,l+1)} = \left(w_{1,k}^{(l,l+1)}, \dots, w_{q^{(l)},k}^{(l,l+1)}\right)^\top \in \mathbb{R}^{q^{(l)}} \tag{A.5}$$

for all $k \in \{1, \dots, q^{(l+1)}\}$.

For optimisation tasks, statisticians usually need to maximise a unique objective function tailored to the specific problem. For neural networks, the common approach is to minimise a loss function with respect to the weights. Hence, the goal of training is to find the neural network's weights $\boldsymbol{w}^* \in \mathcal{W}$ such that

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w} \in \mathcal{W}} \mathbb{E}_{(\boldsymbol{X},Y)}[\mathcal{L}_{\text{FNN}}(\boldsymbol{w}, (\boldsymbol{X}, Y))] \approx \arg\min_{\boldsymbol{w} \in \mathcal{W}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\text{FNN}}(\boldsymbol{w}, (\boldsymbol{x}_i, y_i)) \tag{A.6}$$

where $\mathcal{L}_{\text{FNN}}(\cdot, \cdot)$ is a user-defined loss function that would typically be chosen with consideration of the objectives of the modeller. The model parameters are updated iteratively through mini-batch gradient descent, a process distinct from but complementary to the backpropagation algorithm detailed in LeCun et al. (1998). However, in practice, we usually minimise the loss on the training set and monitor the validation loss for early stopping to avoid overfitting. Early stopping will allow training the neural network for as long as the validation loss is still improving after a pre-specified number of epochs (threshold).

A.4. Mixture density network (MDN)

The MDN (Bishop, 1994) is another parametric deep-learning approach for distributional forecasting in actuarial science. The MDN assumes the true underlying distribution to be a mixture of distributions with K components. The outputs of the MDN consist of parameters of the pre-defined mixture distribution. Let K denote the number of mixture components and $\boldsymbol{\theta}^{(k)}(\boldsymbol{x}; \boldsymbol{w}_{\text{MDN}}) = \left(\theta_1^{(k)}(\boldsymbol{x}; \boldsymbol{w}_{\text{MDN}}), \dots, \theta_{|\boldsymbol{\theta}|}^{(k)}(\boldsymbol{x}; \boldsymbol{w}_{\text{MDN}})\right)^\top \in \mathbb{R}^{|\boldsymbol{\theta}|}$ denote the distributional parameters of the k th component for all $k \in \{1, \dots, K\}$. The MDN $\mathcal{M}_{\boldsymbol{w}_{\text{MDN}}} : \mathbb{R}^p \to \mathbb{R}^{K+K\cdot|\boldsymbol{\theta}|}$ outputs:

$$\mathcal{M}_{\boldsymbol{w}_{\text{MDN}}}(\boldsymbol{x}) = (\boldsymbol{\pi}(\boldsymbol{x}; \boldsymbol{w}_{\text{MDN}})^\top, \boldsymbol{\theta}^{(1)}(\boldsymbol{x}; \boldsymbol{w}_{\text{MDN}})^\top, \dots, \boldsymbol{\theta}^{(K)}(\boldsymbol{x}; \boldsymbol{w}_{\text{MDN}})^\top)^\top, \tag{A.7}$$

where $\boldsymbol{\pi}(\boldsymbol{x}; \boldsymbol{w}_{\text{MDN}}) = (\pi_1(\boldsymbol{x}; \boldsymbol{w}_{\text{MDN}}), \dots, \pi_K(\boldsymbol{x}; \boldsymbol{w}_{\text{MDN}}))^\top \in \mathbb{R}^K$ represents the distribution components' mixing weights. To find the weights $\boldsymbol{w}_{\text{MDN}}$, we minimise the following negative log-likelihood (NLL) function

$$\mathcal{L}_{\text{MDN}}(\boldsymbol{w}, \mathcal{D}) = -\sum_{i=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k(\boldsymbol{x}_i; \boldsymbol{w}) \cdot f_{Y|\boldsymbol{X}}(y_i; \boldsymbol{\theta}^{(k)}(\boldsymbol{x}_i; \boldsymbol{w}))\right) \tag{A.8}$$

with respect to the weights. Mini-batch gradient descent is implemented to minimise the NLL given by Eq. (A.8). In actuarial science, Al-Mudafer et al. (2022) employs the Gaussian MDN for loss reserving, and Delong et al. (2021) examines the effectiveness of a gamma MDN in modelling claim severity.

A.5. (Deep) quantile regression

Key risk measures can be quantified using quantile regression, which is a semi-parametric approach first introduced by Koenker and Bassett (1978). Taylor (2000) proposed deep quantile regression, a deep learning variant of quantile regression models to estimate conditional densities. The deep quantile regression for the α quantile $\mathcal{M}_{\boldsymbol{w}_{\alpha-\text{DQR}}}: \mathbb{R}^p \to \mathbb{R}$ follows

$$\mathcal{M}_{\boldsymbol{w}_{\alpha-\text{DQR}}}(\boldsymbol{x}) = Q_{Y|\boldsymbol{X}}(\alpha|\boldsymbol{x}; \boldsymbol{w}_{\alpha-\text{DQR}}). \tag{A.9}$$

The loss function is the pinball loss, defined as follows:

$$\mathcal{L}_{\alpha-\text{DQR}}(\boldsymbol{w}, \mathcal{D}) = \sum_{i=1}^{N} (y_i - Q_{Y|\boldsymbol{X}}(\alpha|\boldsymbol{x}; \boldsymbol{w}))\big(\alpha - \mathbb{1}_{\{y_i \le Q_{Y|\boldsymbol{X}}(\alpha|\boldsymbol{x};\boldsymbol{w})\}}\big). \tag{A.10}$$

Appendix B. Partitioning algorithm

Algorithm 1 Partitioning Algorithm: Merge Cutpoints.

Input: Initial list of cutpoints $L_{\text{Raw}} = \{c_0^*, \dots, c_{\tilde{K}}^*\}$, training dataset $\mathcal{D}_{\text{Train}}$, minimum observations M.
Output: Refined list of cutpoints L_{Merged} ensuring at least M observations per interval.
Initialise $L_{\text{Merged}} \leftarrow \{c_0^*\}$ and `left` $\leftarrow 0$.
for `right` $\leftarrow 1$ to $\tilde{K} - 1$ **do**
    Count the number of observations $y_i \in \mathcal{D}_{\text{Train}}$ within $[c^*_{\texttt{left}}, c^*_{\texttt{right}})$.
    Count the number of observations $y_i \in \mathcal{D}_{\text{Train}}$ within $[c^*_{\texttt{right}}, c_{\tilde{K}})$.
    if both counts are $\ge M$ **then**
        Append $c^*_{\texttt{right}}$ to L_{Merged} and update `left` $\leftarrow$ `right`.
Append $c^*_{\tilde{K}}$ to L_{Merged}.
return L_{Merged}

Appendix C. Regularisation Dataset Generation

The synthetic dataset for the regularisation study in Section 4.2.2 consists of 40,000 independent and identically distributed (i.i.d.) samples from the random vector $(\boldsymbol{X}, Y)$, with the feature vector $\boldsymbol{X} = (X_1, X_2)^\top \in \mathbb{R}^2$. Each feature X_j is independently distributed according to a Gaussian distribution $\mathcal{N}(0, 0.5^2)$ for $j \in \{1, 2\}$. The conditional distribution of Y given $\boldsymbol{X}$ is also Gaussian, defined as $Y|\boldsymbol{X} \sim \mathcal{N}(-X_1 + X_2, (0.5 \cdot (X_1^2 + X_2^2))^2)$. For this study, the employed baseline model is linear regression, while the DRN adopts a three-hidden-layer architecture with 128 neurons each, a dropout rate of 0.1, a learning rate of 2.5×10^{-4}, and a batch size of 128. The cutpoint ratio is 0.01, with 5 minimum observations required in each partitioned interval. The data is split into 60%, 20%, and 20% for training, validation, and testing, respectively. An early stopping mechanism, with a patience parameter of 20 epochs, is implemented to prevent overfitting. This means the training ceases if validation loss has not improved within 20 epochs.

Appendix D. Baseline Dependency: Simulation Study

This appendix complements Section 5.4 by further examining the DRN's baseline dependency. The DRN effectively treats the baseline model as a distributional reference. Intuitively, the forward–KL penalty in (13) acts on the mass of probability situated in the refinement region, pushing the model towards the baseline distribution *"shape"*, whereas the mean penalty in (15) pushes the model's *"location"* towards the baseline conditional mean. In this sense, different choices of functional penalties and discrepancy are possible, such as reverse KL or Wasserstein distance, and will determine which aspects of the baseline the DRN is encouraged to imitate. This viewpoint fits within the general framework of penalised population-risk minimisation; see, for example, Wegkamp and Yuan (2011) and Murphy (2022). In practice, when the training objective includes a forward–KL regulariser, a well-chosen baseline acts as a reliable and stable anchor whose main distributional patterns are preserved, while the neural network focuses on correcting local misspecification.

Table D.4
Neural Network Architecture and Training Settings.

Hyperparameter | Training Size	3000	6000	12000
Hidden Layers	4	4	4
Hidden Size	256	256	256
Batch Size	128	128	128
Learning Rate	2.5×10^{-4}	2.5×10^{-4}	2.5×10^{-4}
Patience	20	20	20
[a] Proportion	0.2	0.1	0.05
[b] KL α_1 and Roughness α_2	2×10^{-3}	1×10^{-3}	5×10^{-4}

[a] The Proportion column applies to DDR and DRN models.
[b] The KL and roughness penalties only apply to the DRN model.

Here, we also provide the full details of the baseline-dependency simulation study summarised in Section 5.4. We conduct the study to investigate the effect of baseline misspecification on DRN. The aims are twofold: (i) examine whether the quality (from a distributional perspective) of the baseline model impacts downstream DRN performance when compared with competing distributional regression networks such as DDR and MDN, thereby guiding baseline-selection strategies; and (ii) investigate the generalisation capacity of DRN when trained over a poorly specified baseline, particularly compared with other baseline-dependent approaches such as CANN.

For the first aim, we evaluate DRN under varying distributional assumptions and model complexities for the GLM baseline (Gamma vs. Gaussian, full vs. null) and compare its performance with DDR and MDN across training-set sizes. For the second aim, we study how DRN and CANN respond to different baseline choices under a fixed distributional assumption. More specifically, we begin by fitting GLM baselines under varying distributional assumptions and model assumptions, i.e., full vs null models. Here, "full" refers to the usual GLM (Nelder and Wedderburn, 1972) with an intercept and both covariates in the linear predictor and a single parameter vector shared across all observations. The "null" model is the corresponding intercept-only GLM with no covariates. In particular, we are not using a "full" model in the sense of having one parameter per observation. Subsequently, we train CANNs and DRNs using each of these GLM baselines, while DDRs are trained independently without reliance on a baseline. Each model is optimised using its respective loss function: CANNs use deviance or NLL loss, DDRs employ JBCE loss, and DRNs adopt the regularised JBCE loss formulation introduced in Section 4.2.2. Model performance is assessed using key evaluation metrics: NLL, CRPS, RMSE, and 90% QL.

Note that we adopt a prudent yet consistent selection of hyperparameters that balances fairness and complexity considerations (but we do not further tune them). Given that the data-generating mechanism is fully specified, we can simulate an arbitrarily large test sample. To reduce Monte Carlo noise in the evaluation of competing models, we fix the test set size at 100,000 observations. Each training set is independently drawn using 20 different initialised seeds.

Hyperparameter and Training Settings All neural networks have 4 hidden layers, where the number of neurons is set to 256. The learning rate, batch size, and patience are fixed at 2.5×10^{-4}, 128, and 20, respectively. We evaluate models across three training set sizes: $\{3000, 6000, 12000\}$. The forward KL penalty parameter α_1 in Eq. (13) and roughness penalty parameter α_2 in Eq. (14) for DRNs are set at 2×10^{-3}, 1×10^{-3} and 5×10^{-4}, for training dataset sizes of 3000, 6000, and 12000, respectively. For DDRs and DRNs, the cutpoint ratio is set to 0.2 for a training size of 3000 and 0.1 for size of 6000 and 0.05 for size of 12000, ensuring a minimum of 5 observations per interval. DRNs trained on Gaussian GLM baselines include an additional cutpoint at −15

Table D.5
Definition of GLMs, including their distributional assumption of $Y|\boldsymbol{X}$, mean function μ, and dispersion parameter ϕ. We define $\mu(\boldsymbol{x};\boldsymbol{\beta}) = \mathbb{E}[Y|\boldsymbol{x};\boldsymbol{\beta}] = g^{-1}([1,\boldsymbol{x}^\top]\boldsymbol{\beta})$, $g(x) = \log(x)$ for all GLMs.

Model	Distribution	$\boldsymbol{\beta}^*$	$\hat{\phi}$
GLM_GAMMA	Gamma	$\arg\min_{\boldsymbol{\beta}} \text{Deviance}_{\text{GAMMA}}(\mathcal{D},\boldsymbol{\beta})$,	$\frac{1}{n-3}\sum_{i=1}^{n}\frac{(y_i-\mu(\boldsymbol{x}_i;\boldsymbol{\beta}^*))^2}{\mu(\boldsymbol{x}_i;\boldsymbol{\beta}^*)^2}$
GLM_GAUSSIAN	Gaussian	$\arg\min_{\boldsymbol{\beta}} \text{Deviance}_{\text{GAUSSIAN}}(\mathcal{D},\boldsymbol{\beta})$	$\frac{1}{n-3}\sum_{i=1}^{n}(y_i-\mu(\boldsymbol{x}_i;\boldsymbol{\beta}^*))^2$
GLM_GAMMA_NULL	Gamma	$[g(\sum_{i=1}^{n} y_i/n), 0, 0]^\top$	$\frac{1}{n-3}\sum_{i=1}^{n}\frac{(y_i-\mu(\boldsymbol{x}_i;\boldsymbol{\beta}^*))^2}{\mu(\boldsymbol{x}_i;\boldsymbol{\beta}^*)^2}$
GLM_GAUSSIAN_NULL	Gaussian	$[g(\sum_{i=1}^{n} y_i/n), 0, 0]^\top$	$\frac{1}{n-3}\sum_{i=1}^{n}(y_i-\mu(\boldsymbol{x}_i;\boldsymbol{\beta}^*))^2$

compared with those trained on Gamma GLMs. A summary of the neural network architecture and training settings is provided in Table b.

Data Generating Process The feature vectors follow a multivariate Gaussian distribution, and the response is a sum of a Gamma component and a transformed Gaussian component:

$$\begin{aligned}\boldsymbol{X} &= (X_1, X_2)^\top \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),\\ G|\boldsymbol{X} &\sim \text{Gamma}\big(\mu_G(\boldsymbol{X}), \phi_G(\boldsymbol{X})\big),\\ Z|\boldsymbol{X} &\sim \mathcal{N}\big(\mu_Z(\boldsymbol{X}), \sigma_Z^2(\boldsymbol{X})\big),\\ Y &= G + |Z|^3,\end{aligned} \tag{D.1}$$

where

- the features exhibit positive correlation, defined by

$$\boldsymbol{\mu} = (0,0)^\top, \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.25 \\ 0.25 & 1 \end{bmatrix}, \tag{D.2}$$

- and the mean/dispersion functions for the Gamma component and the mean/scale functions for the Gaussian component are

$$\mu_G(\boldsymbol{X}) = \exp\left(-\frac{X_1}{2} + \frac{X_2}{2}\right) + \tfrac{1}{2}\left|\sin\big(\pi(X_1+X_2)\big)\right|, \tag{D.3}$$

$$\phi_G(\boldsymbol{X}) = \frac{\exp(X_1/3)}{1+\exp\big(X_1X_2\big)}, \tag{D.4}$$

$$\mu_Z(\boldsymbol{X}) = -\frac{X_1}{2} + \frac{X_2}{2} + \sin\big(\pi(X_1+X_2)\big), \tag{D.5}$$

$$\sigma_Z(\boldsymbol{X}) = \frac{\exp\big(\frac{1}{2}X_1X_2\big)}{1+\exp\big(X_1X_2\big)}. \tag{D.6}$$

Table D.5 defines all the baseline GLMs. Specifically, `GLM_GAMMA` and `GLM_GAUSSIAN` assume Gamma and Gaussian distributions, respectively, and are fully specified models with coefficient vectors and dispersion parameters estimated from the training data. Within each distributional assumption, we include a null model.

The top panel of Table D.6 reports the performance (on the common test set of 100,000 simulated observations) of all GLMs across 20 seeds and three training sizes. Two clear patterns emerge: (i) Gamma GLMs consistently outperform Gaussian GLMs in NLL, CRPS, and RMSE; and (ii) the null variants perform worse than their corresponding full models, as expected. The Gaussian GLMs are deliberately strongly misspecified in this experiment: they fit a symmetric distribution on the entire real line to a strictly positive and highly skewed response, see Eq. (D.1). As a result, they allocate non-zero probability mass to negative values and systematically under-represent the right tail, which leads to noticeably larger NLL and CRPS values than the corresponding Gamma GLMs for all sample sizes. In that sense, the Gaussian family is a "bad" baseline for this data-generating process, both conceptually and empirically.

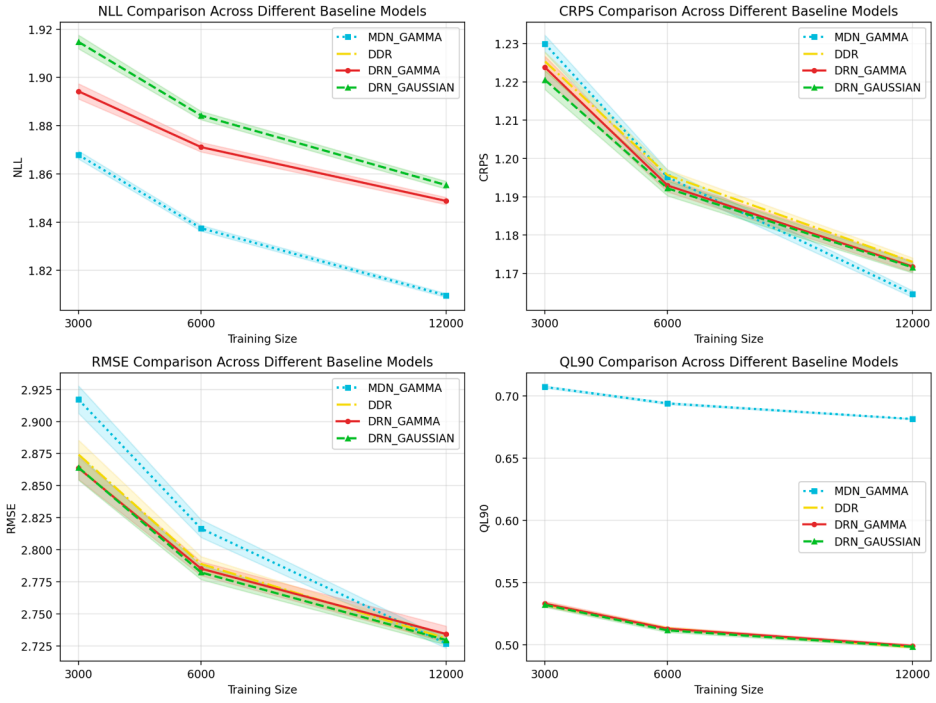
The middle panel displays the performance of DRNs trained on the baseline GLMs from the top panel. A comparison between the top and bottom panels indicates that CANNs largely inherit the ordering of their baseline models. By contrast, the DRNs in the middle panel demonstrate substantially reduced variation across baselines, highlighting their robustness to baseline misspecification.

Table D.7 and Fig. D.12 summarise how DRN compares with the other flexible distributional regression models across training-set sizes. Table D.7 reports the mean (± standard deviation) of NLL, CRPS, RMSE and QL90 for the Gamma MDN, DDR and the two DRN variants, while Fig. D.12 presents the same information graphically as learning curves: the top panel contrasts DRN with DDR and the Gamma MDN, and the bottom panel repeats the comparison for CANN. The top panel of Fig. D.12 and Table D.7 show that DRN is remarkably robust in this setting: even with a Gaussian baseline, `DRN_GAUSSIAN` tracks `DRN_GAMMA` very closely and in fact attains the better CRPS, RMSE and QL90 across training sizes, and only slightly worse NLL. When compared to other flexible distributional regression models, DRN clearly dominates DDR in most cases (better CRPS, RMSE and finite NLL, and highly competitive QL90 across training sizes), and DRNs even outperform the more flexible Gamma MDNs for smaller datasets in terms of CRPS and RMSE, while being uniformly superior to Gamma MDNs in QL90 but consistently inferior in NLL. Overall, these results indicate that, even under deliberate and severe baseline misspecification (assuming $Y|\boldsymbol{x}$ to be Gaussian when it is actually generated as a strictly positive, highly skewed variable), DRN does not suffer a substantial loss of predictive accuracy and is in fact often competitive with or better than DDR, CANN and Gamma MDNs, alleviating concerns about excessive baseline dependence. CANN, on the other hand, largely inherits the shape of its baseline as shown in the bottom panel of Fig. D.12. `CANN_GAUSSIAN` is consistently worse than `CANN_GAMMA`, which is expected because CANN only adjusts the mean while keeping the misspecified distributional form fixed.

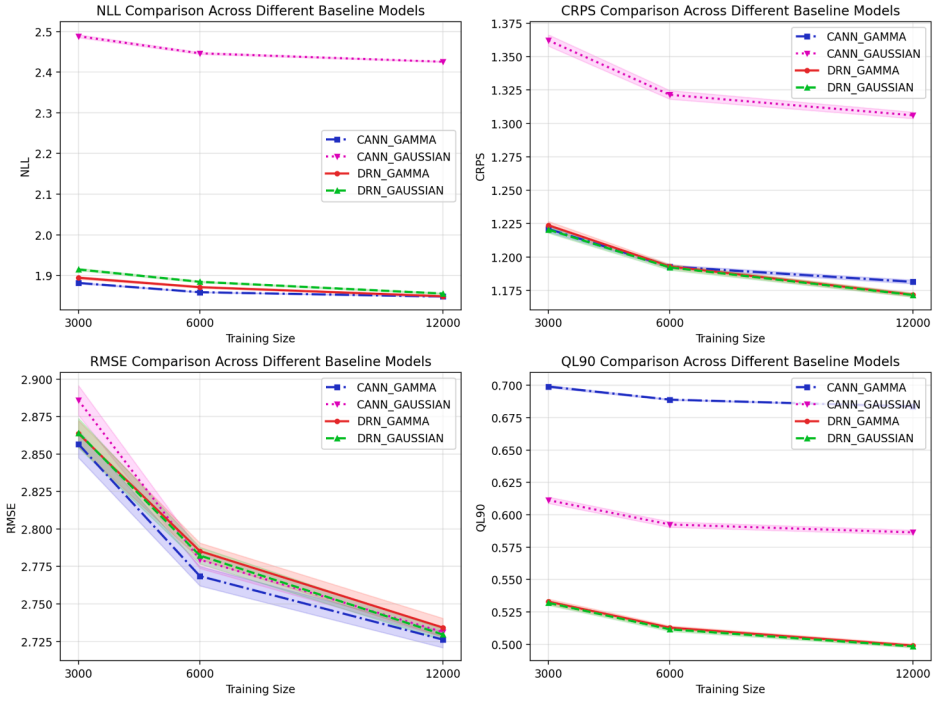
In conclusion, we have varied the baseline distributional assumptions, model complexity and training-set sizes, and compared DRN with DDR, MDN and CANN. The results show that DRN is generally robust to the choice of baseline: with reasonably specified baselines, it consistently improves upon the GLM and typically matches or outperforms DDR and Gamma MDNs in CRPS, RMSE and QL90, especially for smaller samples. It is worth noting that, even with deliberately misspecified Gaussian baselines—fitting a symmetric distribution on the real line to a strictly positive, highly skewed response—DRN still attains competitive scores and remains markedly better than CANN, which largely inherits the weaknesses of its baseline. In other words, a good baseline is desirable and helps the regularisation behave as intended. Still, even when the baseline is imperfect, the DRN can correct substantial misspecification rather than being dragged down by it.

Table D.6
Performance on the common test set of 100,000 simulated observations for baseline GLMs, DRNs, and CANNs. Each panel reports the mean ± standard deviation of NLL, CRPS, RMSE, and QL90 across 20 seeds for training set sizes 3,000, 6,000, and 12,000. The top panel shows the baseline GLMs, the middle panel the DRNs trained on each GLM baseline, and the bottom panel the corresponding CANNs. Lower values indicate better performance.

GLMs				
Model	Metric	3000	6000	12000
GLM_GAMMA	NLL	**2.3149 ± 0.0038**	**2.3104 ± 0.0024**	**2.3116 ± 0.0022**
GLM_GAMMA_NULL	NLL	2.3162 ± 0.0035	2.3135 ± 0.0018	2.3139 ± 0.0015
GLM_GAUSSIAN	NLL	2.8264 ± 0.0007	2.8242 ± 0.0002	2.8238 ± 0.0002
GLM_GAUSSIAN_NULL	NLL	2.8472 ± 0.0006	2.8453 ± 0.0002	2.8451 ± 0.0002
GLM_GAMMA	CRPS	**1.7849 ± 0.0019**	**1.7813 ± 0.0016**	**1.7820 ± 0.0012**
GLM_GAMMA_NULL	CRPS	1.8206 ± 0.0014	1.8193 ± 0.0007	1.8200 ± 0.0005
GLM_GAUSSIAN	CRPS	1.9049 ± 0.0047	1.8986 ± 0.0022	1.8989 ± 0.0017
GLM_GAUSSIAN_NULL	CRPS	1.9570 ± 0.0047	1.9525 ± 0.0021	1.9535 ± 0.0016
GLM_GAMMA	RMSE	**4.0733 ± 0.0032**	**4.0688 ± 0.0021**	**4.0680 ± 0.0013**
GLM_GAMMA_NULL	RMSE	4.1618 ± 0.0003	4.1611 ± 0.0001	4.1610 ± 0.0001
GLM_GAUSSIAN	RMSE	4.0767 ± 0.0008	4.0739 ± 0.0002	4.0732 ± 0.0001
GLM_GAUSSIAN_NULL	RMSE	4.1618 ± 0.0003	4.1611 ± 0.0001	4.1610 ± 0.0001
GLM_GAMMA	QL90	0.9967 ± 0.0024	0.9946 ± 0.0012	0.9950 ± 0.0010
GLM_GAMMA_NULL	QL90	0.9998 ± 0.0012	0.9981 ± 0.0005	0.9982 ± 0.0004
GLM_GAUSSIAN	QL90	**0.9844 ± 0.0014**	**0.9818 ± 0.0007**	**0.9818 ± 0.0005**
GLM_GAUSSIAN_NULL	QL90	1.0012 ± 0.0014	0.9992 ± 0.0006	0.9993 ± 0.0005
		DRNs		
DRN_GAMMA	NLL	1.8942 ± 0.0031	1.8711 ± 0.0020	**1.8488 ± 0.0015**
DRN_GAMMA_NULL	NLL	**1.8936 ± 0.0031**	**1.8694 ± 0.0021**	1.8490 ± 0.0014
DRN_GAUSSIAN	NLL	1.9147 ± 0.0028	1.8842 ± 0.0017	1.8553 ± 0.0016
DRN_GAUSSIAN_NULL	NLL	1.9106 ± 0.0031	1.8785 ± 0.0019	1.8567 ± 0.0017
DRN_GAMMA	CRPS	1.2238 ± 0.0029	1.1929 ± 0.0017	1.1718 ± 0.0014
DRN_GAMMA_NULL	CRPS	1.2228 ± 0.0023	1.1911 ± 0.0017	1.1725 ± 0.0013
DRN_GAUSSIAN	CRPS	**1.2205 ± 0.0025**	1.1922 ± 0.0020	**1.1715 ± 0.0014**
DRN_GAUSSIAN_NULL	CRPS	1.2229 ± 0.0033	**1.1910 ± 0.0021**	1.1727 ± 0.0015
DRN_GAMMA	RMSE	**2.8636 ± 0.0088**	2.7852 ± 0.0054	2.7343 ± 0.0062
DRN_GAMMA_NULL	RMSE	2.8657 ± 0.0098	2.7873 ± 0.0083	2.7341 ± 0.0045
DRN_GAUSSIAN	RMSE	2.8641 ± 0.0098	2.7823 ± 0.0055	**2.7296 ± 0.0036**
DRN_GAUSSIAN_NULL	RMSE	2.8716 ± 0.0134	**2.7808 ± 0.0069**	2.7315 ± 0.0038
DRN_GAMMA	QL90	0.5330 ± 0.0020	0.5129 ± 0.0014	0.4991 ± 0.0012
DRN_GAMMA_NULL	QL90	**0.5320 ± 0.0020**	0.5129 ± 0.0016	0.4994 ± 0.0009
DRN_GAUSSIAN	QL90	0.5320 ± 0.0018	0.5116 ± 0.0015	**0.4985 ± 0.0010**
DRN_GAUSSIAN_NULL	QL90	0.5340 ± 0.0025	**0.5107 ± 0.0013**	0.4991 ± 0.0009
		CANNs		
CANN_GAMMA	NLL	**1.8814 ± 0.0017**	**1.8587 ± 0.0008**	**1.8480 ± 0.0008**
CANN_GAMMA_NULL	NLL	1.8828 ± 0.0022	1.8593 ± 0.0011	1.8487 ± 0.0011
CANN_GAUSSIAN	NLL	2.4883 ± 0.0037	2.4463 ± 0.0024	2.4255 ± 0.0012
CANN_GAUSSIAN_NULL	NLL	2.4912 ± 0.0038	2.4495 ± 0.0022	2.4263 ± 0.0009
CANN_GAMMA	CRPS	**1.2210 ± 0.0022**	**1.1928 ± 0.0013**	**1.1814 ± 0.0014**
CANN_GAMMA_NULL	CRPS	1.2231 ± 0.0026	1.1942 ± 0.0017	1.1821 ± 0.0015
CANN_GAUSSIAN	CRPS	1.3621 ± 0.0043	1.3214 ± 0.0033	1.3061 ± 0.0024
CANN_GAUSSIAN_NULL	CRPS	1.3631 ± 0.0048	1.3230 ± 0.0034	1.3086 ± 0.0029
CANN_GAMMA	RMSE	**2.8566 ± 0.0091**	**2.7686 ± 0.0065**	**2.7262 ± 0.0054**
CANN_GAMMA_NULL	RMSE	2.8737 ± 0.0147	2.7975 ± 0.0189	2.7319 ± 0.0070
CANN_GAUSSIAN	RMSE	2.8858 ± 0.0099	2.7795 ± 0.0058	2.7313 ± 0.0032
CANN_GAUSSIAN_NULL	RMSE	2.8938 ± 0.0103	2.7892 ± 0.0056	2.7331 ± 0.0024
CANN_GAMMA	QL90	0.6990 ± 0.0009	0.6889 ± 0.0004	0.6837 ± 0.0005
CANN_GAMMA_NULL	QL90	0.6998 ± 0.0012	0.6888 ± 0.0005	0.6841 ± 0.0005
CANN_GAUSSIAN	QL90	**0.6113 ± 0.0025**	**0.5925 ± 0.0021**	0.5864 ± 0.0018
CANN_GAUSSIAN_NULL	QL90	0.6138 ± 0.0033	0.5951 ± 0.0018	**0.5859 ± 0.0015**



(a) Comparison of DRN trained with different baseline assumptions vs DDR and MDN.



(b) Comparison of DRN vs CANN when trained with different baseline assumptions.

Fig. D.12. Performance of DRN compared to DDR and MDN (top panel) and CANN (bottom panel) across CRPS, RMSE, NLL, and QL90. Each subplot represents a different metric, with solid lines denoting mean performance and shaded regions indicating standard deviation over 20 training runs.

Table D.7
Comparison of DRN vs DDR vs MDN. Columns represent training set sizes (3000, 6000, 12000), and rows show models evaluated across four metrics (CRPS, RMSE, NLL, and QL90). Lower values indicate better performance, with the best-performing model in each setting highlighted in bold. Results align with the top panel of Fig. D.12, visualising trends across 20 training seeds.

MDN vs DDR vs DRN				
Model	Metric	3000	6000	12000
MDN_GAMMA	NLL	**1.8679 ± 0.0017**	**1.8375 ± 0.0012**	**1.8096 ± 0.0008**
DDR	NLL	∞	∞	∞
DRN_GAMMA	NLL	1.8942 ± 0.0031	1.8711 ± 0.0020	1.8488 ± 0.0015
DRN_GAUSSIAN	NLL	1.9147 ± 0.0028	1.8842 ± 0.0017	1.8553 ± 0.0016
MDN_GAMMA	CRPS	1.2299 ± 0.0023	1.1951 ± 0.0019	**1.1646 ± 0.0009**
DDR	CRPS	1.2253 ± 0.0025	1.1956 ± 0.0014	1.1730 ± 0.0011
DRN_GAMMA	CRPS	1.2238 ± 0.0029	1.1929 ± 0.0017	1.1718 ± 0.0014
DRN_GAUSSIAN	CRPS	**1.2205 ± 0.0025**	**1.1922 ± 0.0020**	1.1715 ± 0.0014
MDN_GAMMA	RMSE	2.9170 ± 0.0107	2.8165 ± 0.0069	**2.7269 ± 0.0036**
DDR	RMSE	2.8742 ± 0.0112	2.7889 ± 0.0058	2.7308 ± 0.0033
DRN_GAMMA	RMSE	**2.8636 ± 0.0088**	2.7852 ± 0.0054	2.7343 ± 0.0062
DRN_GAUSSIAN	RMSE	2.8641 ± 0.0098	**2.7823 ± 0.0055**	2.7296 ± 0.0036
MDN_GAMMA	QL90	0.7071 ± 0.0008	0.6938 ± 0.0008	0.6813 ± 0.0004
DDR	QL90	0.5324 ± 0.0017	0.5132 ± 0.0012	**0.4981 ± 0.0007**
DRN_GAMMA	QL90	0.5330 ± 0.0020	0.5129 ± 0.0014	0.4991 ± 0.0012
DRN_GAUSSIAN	QL90	**0.5320 ± 0.0018**	**0.5116 ± 0.0015**	0.4985 ± 0.0010

Appendix E. Activation functions

The Leaky Rectified Linear Unit (`LeakyReLU`) is recommended as the activation function of choice between the input, the first hidden layer, and between hidden layers. Other choices, such as the Rectified Linear Unit (`ReLU`) and the hyperbolic tangent function (`tanh`) function, can also be viable. The preference for `LeakyReLU` serves as a remedy for the "Dying ReLU" issue encountered with the standard `ReLU` activation function (Alzubaidi et al., 2021). In scenarios where `ReLU` is employed, neurons may cease learning altogether due to an absence of gradient descent updates when their outputs are consistently zero. `LeakyReLU` addresses this by introducing a small, positive gradient for negative inputs.

The `Linear` activation function is selected for the output layer, promoting model flexibility and reducing numerical instability. This is highly recommended to maximise model flexibility and minimise numerical instability. While alternatives like the exponential function (`exp`) and the hyperbolic tangent function (`tanh`) are theoretically viable, they are generally not recommended. The use of the exponential function can potentially result in exploding gradients, and the hyperbolic tangent may fail to achieve optimal refinement.

Appendix F. Hyperparameters for the numerical examples

We briefly summarise the Gaussian Process (GP)-based Bayesian optimisation strategy employed for hyperparameter tuning. Let $\mathcal{L}(\mathcal{D}, \boldsymbol{h})$ denote the loss function evaluated on dataset $\mathcal{D}$ given a hyperparameter configuration $\boldsymbol{h}$. The procedure begins with a set of N_0 randomly selected hyperparameter settings (referred to as "random initialisations" in the main text), for which the corresponding loss values are computed:

$$\{(\boldsymbol{h}_i, \mathcal{L}(\mathcal{D}, \boldsymbol{h}_i))\}_{i=1}^{N_0}.$$

These evaluations define the initial training data for constructing the GP surrogate model:

$$\mathcal{L}_{N_0}(\boldsymbol{h}) \sim \mathcal{GP}\Big(\boldsymbol{m}\Big(\boldsymbol{h}|\{(\boldsymbol{h}_i, \mathcal{L}(\mathcal{D}, \boldsymbol{h}_i))\}_{i=1}^{N_0}\Big), \boldsymbol{K}\Big(\boldsymbol{h}, \boldsymbol{h}'|\{(\boldsymbol{h}_i, \mathcal{L}(\mathcal{D}, \boldsymbol{h}_i))\}_{i=1}^{N_0}\Big)\Big).$$

The model is then iteratively refined by selecting additional hyperparameter settings that maximise an acquisition function, such as Expected Improvement (EI). For $j = N_0, \ldots, N-1$, the next point is selected via:

$$\boldsymbol{h}_{j+1} = \arg\max_{\boldsymbol{h}} \mathrm{EI}\Big(\boldsymbol{h}, \mathcal{L}_j, \{(\boldsymbol{h}_i, \mathcal{L}(\mathcal{D}, \boldsymbol{h}_i))\}_{i=1}^{j}\Big),$$

$$\mathcal{L}_{j+1}(\boldsymbol{h}) \sim \mathcal{GP}\Big(\boldsymbol{m}\Big(\boldsymbol{h}|\{(\boldsymbol{h}_i, \mathcal{L}(\mathcal{D}, \boldsymbol{h}_i))\}_{i=1}^{j+1}\Big), \boldsymbol{K}\Big(\boldsymbol{h}, \boldsymbol{h}'|\{(\boldsymbol{h}_i, \mathcal{L}(\mathcal{D}, \boldsymbol{h}_i))\}_{i=1}^{j+1}\Big)\Big).$$

Table F.8
(Synthetic Dataset) Hyperparameter Ranges Across Models. $Real(a, b)$ means the hyperparameter space is continuous from a to b (inclusive). $Int(a, b)$ means the hyperparameter space consists of discrete integer values from a to b (inclusive).

Hyperparameter	CANN	MDN	DDR	DRN
Learning Rate	$Real(0.0001, 0.005)$	$Real(0.0001, 0.005)$	$Real(0.0001, 0.005)$	$Real(0.0001, 0.005)$
Batch Size	[128, 256, 512]	[128, 256, 512]	[128, 256, 512]	128
Dropout Rate	$Real(0.0, 0.5)$	$Real(0.0, 0.5)$	$Real(0.0, 0.5)$	0.1
Number of Layers	$Int(1, 4)$	$Int(1, 4)$	$Int(1, 4)$	4
Neurons per Layer	[16, 32, 64, 128, 256, 512]	[16, 32, 64, 128, 256, 512]	[16, 32, 64, 128, 256, 512]	256
Mixture Components	-	$Int(1, 10)$	-	-
Proportion	-	-	[0.025, 0.05]	[0.025, 0.05]
Min.[a]	-	-	-	[1, 3, 5]
KL Penalty	-	-	-	$Real(10^{-4}, 5 \times 10^{-2})$
Roughness Penalty	-	-	-	10^{-3}
Mean Penalty	-	-	-	$Real(10^{-4}, 10^{-1})$

[a] The minimum number of training observations required in each partitioned interval.

Table F.9
(Synthetic Dataset) Hyperparameters for the competing models. The distributional assumption is gamma for CANN and MDN.

Hyperparameters	CANN	MDN	DDR	DRN
Learning Rate	0.00164	0.00039	0.00457	0.00019
Batch Size	512	128	128	128
Dropout Rate	0.48330	0.00232	0.27768	0.10000
Number of Layers	2	4	1	4
Neurons per Layer	256	32	32	256
Mixture Components	-	2	-	-
Cutpoints Ratio (Min.[a])	-	-	0.025 (0)	0.025 (3)
KL Penalty	-	-	-	0.01591
Roughness Penalty	-	-	-	0.00100
Mean Penalty	-	-	-	0.10000
Activation Function	`LeakyReLU`	`LeakyReLU`	`LeakyReLU`	`LeakyReLU`
Output Activation	`exponential`	`Softmax`, `Softplus`	`Softmax`	`Linear`
Hyperparameter Tuning Time [b]	1495 s	2530 s	9963 s	7119 s

[a] The minimum number of training observations required in each partitioned interval.
[b] We trained on a CPU rather than on a GPU, as the networks were relatively small.

Table F.10
Top 4 hyperparameter settings for the DRN model, selected based on validation CRPS.

Hyperparameters	DRN_1	DRN_2	DRN_3	DRN_4
Learning Rate	0.00019	0.00024	0.00034	0.00028
KL Penalty	0.01591	0.00202	0.00011	0.00010
Mean Penalty	0.10000	0.03342	0.10000	0.10000
Roughness Penalty	0.00100	0.00100	0.00100	0.00100
Cutpoints Ratio	0.025	0.025	0.025	0.025
Min. Obs. per Interval	3	5	1	3
Batch Size	128	128	128	128
Dropout Rate	0.1	0.1	0.1	0.1
Number of Layers	4	4	4	4
Neurons per Layer	256	256	256	256
Activation Function	`LeakyReLU`	`LeakyReLU`	`LeakyReLU`	`LeakyReLU`
Output Activation	`Linear`	`Linear`	`Linear`	`Linear`

Note: These settings correspond to the four best-performing DRN configurations ranked by CRPS on the validation set. All models employ a gamma GLM as the baseline.

Table F.11
(`freMPL1` Dataset) Hyperparameter Ranges Across Models. $Real(a, b)$ means the hyperparameter space is continuous from a to b (inclusive). $Int(a, b)$ means the hyperparameter space consists of discrete integer values from a to b (inclusive).

Hyperparameter	CANN/CANN_NULL	MDN	DDR	DRN/DRN_NULL
Learning Rate	$Real(0.0001, 0.01)$	$Real(0.0001, 0.01)$	$Real(0.0002, 0.01)$	$Real(0.0001, 0.01)$
Batch Size	[64, 128, 256, 512]	[64, 128, 256, 512]	[64, 128, 256, 512]	128
Dropout Rate	$Real(0.0, 0.5)$	$Real(0.0, 0.5)$	$Real(0.0, 0.5)$	0.1
Number of Layers	$Int(1, 6)$	$Int(1, 6)$	$Int(1, 6)$	3
Neurons per Layer	[32, 64, 128, 256, 512]	[32, 64, 128, 256, 512]	[32, 64, 128, 256, 512]	256
Mixture Components	-	$Int(2, 10)$	-	-
Proportion	-	-	[0.1, 0.2, 0.3, 0.4]	[0.1, 0.2, 0.3, 0.4]
Min.[a]	-	-	-	[3, 5, 10]
KL Penalty	-	-	-	$Real(10^{-5}, 10^{-1})$
Roughness Penalty	-	-	-	$Real(10^{-5}, 10^{-1})$
Mean Penalty	-	-	-	$Real(10^{-5}, 10^{-1})$

[a] The minimum number of training observations required in each partitioned interval.

Table F.12
(`freMPL1` Dataset) Hyperparameters for the competing models. The distributional assumption is gamma for the CANN and MDN.

Hyperparameters	CANN	CANN_NULL	MDN	DDR	DRN	DRN_NULL
Learning Rate	0.00583	0.00904	0.01000	0.00020	0.01000	0.00075
Batch Size	64	256	64	128	128	128
Dropout Rate	0.42345	0.37935	0.50000	0.0	0.10000	0.10000
Number of Layers	2	5	1	1	3	3
Neurons per Layer	128	256	32	512	256	256
Mixture Components	-	-	2	-	-	-
Cutpoints Ratio (Min.[a])	-	-	-	-	0.3 (10)	0.4 (10)
KL Penalty	-	-	-	-	0.00004	0.00001
Roughness Penalty	-	-	-	-	0.1	0.00004
Mean Penalty	-	-	-	-	0.04851	0.00062
Activation Function	`LeakyReLU`	`LeakyReLU`	`LeakyReLU`	`LeakyReLU`	`LeakyReLU`	`LeakyReLU`
Output Activation	`exponential`	`exponential`	`Softmax, Softplus`	`Softmax`	`Linear`	`Linear`

[a] The minimum number of training observations required in each partitioned interval.

This process continues until a total of N hyperparameter configurations (referred to as "evaluations" in the main text) have been evaluated.

Appendix G. Evaluation Metrics

Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ be the dataset for evaluation and $\boldsymbol{w}$ represent the trained parameters/weights of the model. The four evaluation metrics are continuous ranked probability score (CRPS), negative log-likelihood (NLL), root mean squared error (RMSE), and α quantile loss (α QL), respectively.

1. Continuous ranked probability score (CRPS):

$$\text{CRPS}(\mathcal{D}, \boldsymbol{w}) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{N} \int_y \left(F_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_i; \boldsymbol{w}) - \mathbb{1}_{\{y > y_i\}} \right)^2 \mathrm{d}y. \tag{G.1}$$

   A lower CRPS score indicates better model performance.
2. Negative log-likelihood (NLL):

$$\text{NLL}(\mathcal{D}, \boldsymbol{w}) = -\frac{1}{|\mathcal{D}|} \sum_{i=1}^{N} \ln f_{Y|\boldsymbol{X}}(y_i|\boldsymbol{x}_i; \boldsymbol{w}). \tag{G.2}$$

   A lower NLL score indicates better model performance.
3. Root mean squared error (RMSE):

$$\text{RMSE}(\mathcal{D}, \boldsymbol{w}) = \sqrt{\frac{1}{|\mathcal{D}|} \sum_{i=1}^{N} \left(\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x}_i; \boldsymbol{w}] - y_i \right)^2}. \tag{G.3}$$

   A lower RMSE score indicates a better model performance.
4. α quantile loss (α QL):

$$\text{QL}_\alpha(\mathcal{D}, \boldsymbol{w}) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{N} (y_i - Q_{Y|\boldsymbol{X}}(\alpha|\boldsymbol{x}_i; \boldsymbol{w})) \left(\alpha - \mathbb{1}_{\{y_i \le Q_{Y|\boldsymbol{X}}(\alpha|\boldsymbol{x}_i; \boldsymbol{w})\}} \right). \tag{G.4}$$

   A lower QL indicates better model performance.

Appendix H. Kernel SHAP

The Shapley value of a feature X_j represents the average marginal contribution of X_j to the model's prediction across all possible coalitions of features. Specifically, the Shapley value for the jth feature of instance $\boldsymbol{x}$ is given by

$$\phi_j(v_{\mathcal{M}}, \boldsymbol{x}) = \sum_{S \subseteq D \setminus \{j\}} \frac{|S|!(p - |S| - 1)!}{p!} (v_{\mathcal{M}}(S \cup \{j\}, \boldsymbol{x}) - v_{\mathcal{M}}(S, \boldsymbol{x})), \quad j = 1, \dots, p. \tag{H.1}$$

where p is the number of features and $S \subseteq D = \{1, \dots, p\}$ represents a subset of the feature component indices and $v_{\mathcal{M}}(S, \boldsymbol{x})$ is a value function.

Lundberg and Lee (2017) proposed the Kernel SHAP that solves the following constrained optimisation problem

$$\boldsymbol{\phi}(v_{\mathcal{M}}, \boldsymbol{x}) = \arg\min_{\boldsymbol{\beta}} \sum_{S \subseteq D} (v_{\mathcal{M}}(S, \boldsymbol{x}) - \langle \boldsymbol{z}'(S), \boldsymbol{\beta} \rangle)^2 \cdot \frac{p - 1}{\binom{p}{|S|} |S| \cdot (p - |S|)} \tag{H.2}$$

$$\text{s.t. } \beta_0 = v_{\mathcal{M}}(\emptyset, D) \equiv \mathbb{E}_{\boldsymbol{X}}[\mathcal{M}(\boldsymbol{X})], \ \langle \mathbf{1}, \boldsymbol{\beta} \rangle = v_{\mathcal{M}}(D, \boldsymbol{x}) \equiv \mathcal{M}(\boldsymbol{x}), \tag{H.3}$$

where $\boldsymbol{\phi}(v_{\mathcal{M}}, \boldsymbol{x}) = (\phi_0(v_{\mathcal{M}}, D), \phi_1(v_{\mathcal{M}}, \boldsymbol{x}), \dots, \phi_p(v_{\mathcal{M}}, \boldsymbol{x}))^\top \in \mathbb{R}^{p+1}$, $\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_p)^\top \in \mathbb{R}^{p+1}$ and $\boldsymbol{z}'(S) = (1, \mathbb{1}_{\{1 \in S\}}, \dots, \mathbb{1}_{\{p \in S\}})^\top \in \{0, 1\}^{p+1}$. Intuitively, the Kernel SHAP method finds $\boldsymbol{\phi}(v_{\mathcal{M}}, \boldsymbol{x})$ to explain the prediction for the instance $\boldsymbol{x}$. To compute the value function $v_{\mathcal{M}}(S, \boldsymbol{x})$,

Kernel SHAP leverages the conditional Shapley value that is the expected value of the regression output from the model $\mathcal{M}$ conditional on $\boldsymbol{X}_S = \boldsymbol{x}_S$. Mathematically,

$$\begin{aligned} v_{\mathcal{M}}(S, \boldsymbol{x}) &= \mathbb{E}_{\boldsymbol{X}_{\bar{S}}|\boldsymbol{X}_S}[\mathcal{M}(\boldsymbol{X})|\boldsymbol{X}_S = \boldsymbol{x}_S] \\ &= \mathbb{E}_{\boldsymbol{X}_{\bar{S}}|\boldsymbol{X}_S}[\mathcal{M}(\boldsymbol{X}_S, \boldsymbol{X}_{\bar{S}})|\boldsymbol{X}_S = \boldsymbol{x}_S] \qquad \text{(H.4)} \\ &= \int_{\boldsymbol{x}_S} \mathcal{M}(\boldsymbol{x}_S, \boldsymbol{x}_{\bar{S}}) \cdot f_{\boldsymbol{X}_{\bar{S}}|\boldsymbol{X}_S}(\boldsymbol{x}_{\bar{S}}|\boldsymbol{x}_S) \, \mathrm{d}\boldsymbol{x}_{\bar{S}} \approx \frac{1}{M} \sum_{m=1}^{M} \mathcal{M}\left(\boldsymbol{x}_S, \boldsymbol{x}_{\bar{S}}^{(m)}\right), \qquad \text{(H.5)} \end{aligned}$$

where $\bar{S}$ denotes the complement of S and $\boldsymbol{X}_S$ represents the components of features $\boldsymbol{X}$ in the feature subset S. The Kernel SHAP employs the Monte Carlo approximation to compute the value function since the conditional density of the missing features $\boldsymbol{X}_{\bar{S}} = \boldsymbol{x}_{\bar{S}}$ given the observed features $\boldsymbol{x}_S$ is rarely known. Therefore, it uses the marginal distribution of $\boldsymbol{X}_{\bar{S}}$ obtained from a subset of the training samples, where $\boldsymbol{x}_{\bar{S}}^{(m)}$ for $m \in \{1, \dots, M\}$ are samples from the training data independent of $\boldsymbol{x}_S$.

Kernel SHAP also facilitates a smooth transition from local to global interpretability. After calculating SHAP values for each observation, $\{(\boldsymbol{x}_i, \boldsymbol{\phi}(v_{\mathcal{M}}, \boldsymbol{x}_i))\}_{i=1}^n$, the SHAP feature importance is computed by

$$I_j = \frac{1}{n} \sum_{i=1}^{n} |\phi_j(v_{\mathcal{M}}, \boldsymbol{x}_i)| \tag{H.6}$$

for all $j \in \{1, \dots, p\}$. Theoretically, a larger I_j value indicates greater importance for feature X_j. Two types of visualisations can be generated: the SHAP beeswarm plot and the SHAP dependence plot. The beeswarm plot presents the distribution of SHAP values for all features across various instances. This visualisation aids in identifying key features and their impact on model predictions. The SHAP dependence plot demonstrates the variation of SHAP values for a particular feature across all data points. The overall shape of the plot allows for the inference of a feature's value on the model's predictions. Further, it permits investigation of interaction effects between features, with points $\{(x_{i,j}, \phi_j(v_{\mathcal{M}}, \boldsymbol{x}_i))\}_{i=1}^n$ colour-coded with the values of another feature $X_k, k \neq j$. Illustra-

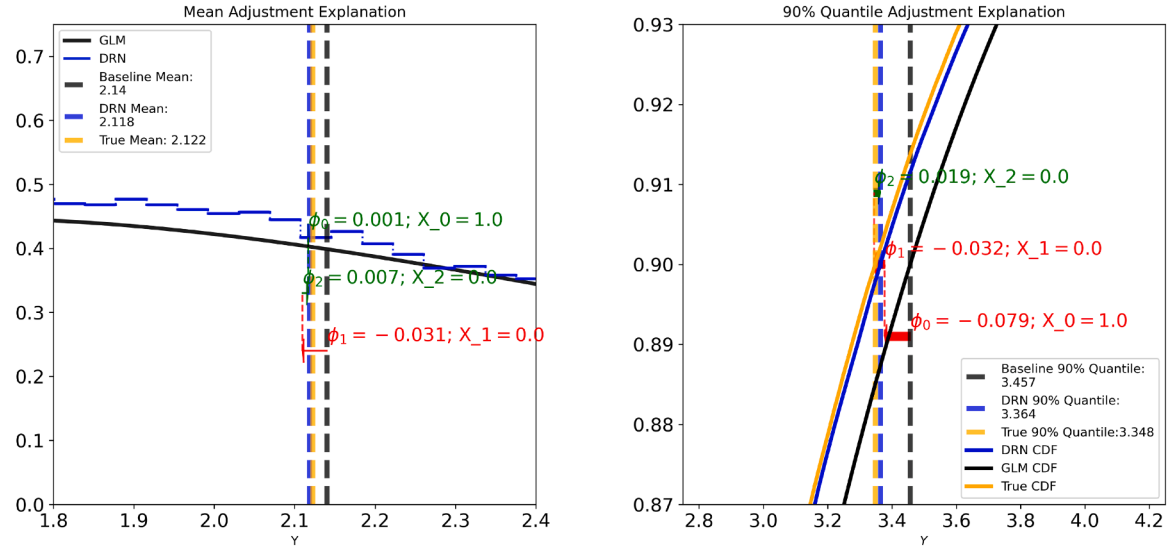


Fig. H.13. The instance of investigation is $\boldsymbol{x}^* = (0.0, 0.0)^\top$. The left panel utilises the Kernel SHAP values to explain the mean adjustment made by the DRN. The right panel utilises the Kernel SHAP values to explain the 90% quantile adjustment made by the DRN. The DRN is decreasing the 90% quantile prediction. The first feature's Kernel SHAP value is $\phi_1 = -0.032$, which implies that it contributes to the 90% quantile prediction 0.032 less in the DRN than within the baseline model. The solid curves represent the estimated density functions: baseline (black), DRN (blue), and true (orange). The vertical dotted lines highlight key distributional properties–means in the left panel and 90% quantiles in the right. Slopes and intersections do not carry inherent meaning. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

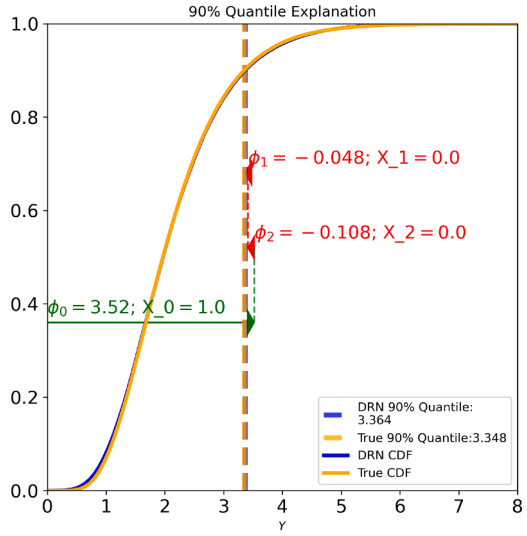


Fig. H.14. This plot utilises the Kernel SHAP values to explain how the DRN predicts the 90% quantile of $Y|\boldsymbol{x}^*$, where $\boldsymbol{x}^* = (0.0, 0.0)^\top$. For example, $\phi_2 = -0.108$ implies that X_2 being 0.0 leads to a 0.108 decrease over the average 90% quantile.

tive examples of these plots are provided in Sections 5.3.2 and H.2.2.

H.1. Synthetic dataset

Building on the preceding example illustrated in Fig. 8, we expect the DRN to (i) perform subtle but correct modifications to the mean predictions and (ii) refine the baseline model's quantile predictions for enhanced accuracy. The first anticipation is exemplified in the left panel of Fig. H.13, which showcases the mean adjustment decomposition utilising the Kernel SHAP values. The mean prediction is closer to the true mean, confirming our initial expectation. The interpretation of how the DRN predicts the mean of $Y|\boldsymbol{x}^*$ is relatively straightforward:

$$\begin{aligned}\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x}^*;\boldsymbol{w},\boldsymbol{\beta}] &= \mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x}^*;\boldsymbol{\beta}] + (\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x}^*;\boldsymbol{w},\boldsymbol{\beta}] - \mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x}^*;\boldsymbol{\beta}]) \\ &= \mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x}^*;\boldsymbol{\beta}] + \phi_0(v_{\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x};\boldsymbol{w},\boldsymbol{\beta}]} - v_{\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x};\boldsymbol{\beta}]}, \mathcal{D}) \\ &\quad + \sum_{j=1}^{2} \phi_j(v_{\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x};\boldsymbol{w},\boldsymbol{\beta}]} - v_{\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x};\boldsymbol{\beta}]}, \boldsymbol{x}^*)\end{aligned}$$

where the "baseline" value ϕ_0, in this context, represents the difference in the average mean predictions between the DRN and the baseline GLM, given by

$$\begin{aligned}&\phi_0(v_{\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x};\boldsymbol{w},\boldsymbol{\beta}]} - v_{\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x};\boldsymbol{\beta}]}, \mathcal{D}) \\ &\quad = \mathbb{E}_{\boldsymbol{X}}[\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{X};\boldsymbol{w},\boldsymbol{\beta}] - \mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{X};\boldsymbol{\beta}]] \\ &\quad = \int_{\boldsymbol{x}} (\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x};\boldsymbol{w},\boldsymbol{\beta}] - \mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x};\boldsymbol{\beta}]) \cdot p_{\boldsymbol{X}}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \\ &\quad \approx \frac{1}{n}\sum_{i=1}^{n} \Big(\mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x}_i;\boldsymbol{w},\boldsymbol{\beta}] - \mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{x}_i;\boldsymbol{\beta}] \Big) \approx 0.001.\end{aligned} \tag{H.7}$$

A value of ϕ_0 close to zero indicates that the mean predictions from DRN and the baseline GLM are nearly identical on average. A positive value of ϕ_j suggests that feature j has a greater contribution to the mean prediction in DRN compared to the baseline.

To verify our second hypothesis, we generate the right panel of Fig. H.13, focusing on adjustments at the 90% quantile level. For this specific instance $\boldsymbol{x}^*$, it is observed that the DRN is correcting the GLM's underestimation of the impacts of X_1 and X_2 ($\phi_1 < 0, \phi_2 > 0$). Specifically,

$$\begin{aligned}Q_{Y|\boldsymbol{X}}(0.9|\boldsymbol{x}^*;\boldsymbol{w},\boldsymbol{\beta}) &\approx Q_{Y|\boldsymbol{X}}(0.9|\boldsymbol{x}^*;\boldsymbol{\beta}) + \phi_0\Big(v_{Q_{Y|\boldsymbol{X}}(0.9|\boldsymbol{x};\boldsymbol{w},\boldsymbol{\beta})} - v_{Q_{Y|\boldsymbol{X}}(0.9|\boldsymbol{x};\boldsymbol{\beta})}\Big) \\ &\quad + \sum_{j=1}^{2} \phi_j(v_{Q_{Y|\boldsymbol{X}}(0.9|\boldsymbol{x};\boldsymbol{w},\boldsymbol{\beta})} - v_{Q_{Y|\boldsymbol{X}}(0.9|\boldsymbol{x};\boldsymbol{\beta})}, \boldsymbol{x}^*) \\ &= Q_{Y|\boldsymbol{X}}(0.9|\boldsymbol{x}^*;\boldsymbol{\beta}) + \underbrace{(-0.079)}_{\phi_0(\text{Right Panel of Figure H.13})}\end{aligned}$$

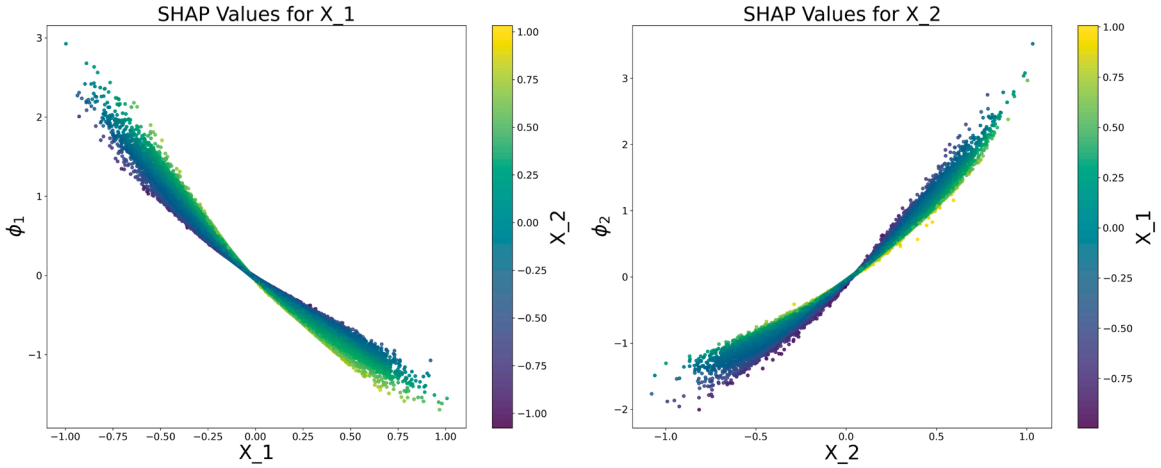


Fig. H.15. SHAP dependence plot for mean predictions made by the DRN. (Left): The graph illustrates how, with an increase in X_1, the DRN model's prediction for the mean decreases relative to the average mean prediction. (Right): Demonstrates that as X_2 decreases, there is a corresponding decrease in the DRN's mean prediction compared to the average, albeit at a diminishing rate.

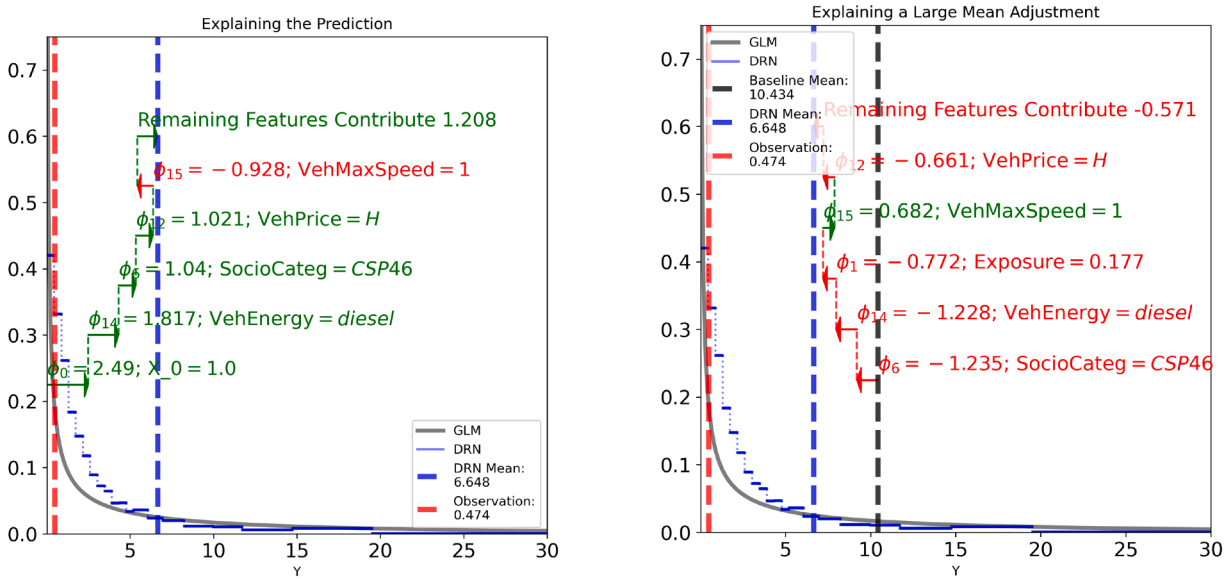


Fig. H.16. The left panel breaks down the DRN's mean prediction for the test-set instance requiring the second-largest mean adjustment, decomposing it from the ground up using SHAP values. The right panel illustrates how the GLM mean (black dotted line) is adjusted upwards to match the DRN predicted mean (blue dotted line), also using SHAP values. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$$+ \underbrace{(-0.032)}_{\phi_1(\text{Right Panel of Figure H.13})} + \underbrace{(+0.019)}_{\phi_2(\text{Right Panel of Figure H.13})} \tag{H.8}$$

On a global level, discrepancies arise between the GLM and the DRN regarding predictions at the 90% quantile level, i.e., there is a noticeable difference in the "baseline" value

$$\phi_0\Big(v_{Q_{Y|\boldsymbol{X}}(0.9|\boldsymbol{x};\boldsymbol{w},\boldsymbol{\beta})} - v_{Q_{Y|\boldsymbol{X}}(0.9|\boldsymbol{x};\boldsymbol{\beta})}\Big) \approx \frac{1}{n}\Bigg(\sum_{i=1}^{n} Q_{Y|\boldsymbol{X}}(0.9|\boldsymbol{x}_i;\boldsymbol{w},\boldsymbol{\beta}) - Q_{Y|\boldsymbol{X}}(0.9|\boldsymbol{x}_i;\boldsymbol{\beta})\Bigg) \approx -0.079$$

implying that the GLM struggles to capture quantile levels accurately, evidenced by the discrepancy in the 90% quantile loss demonstrated in Table 1. The improvement in loss reduction achieved by the DRN emphasises its capacity to overcome the limitations inherent in the baseline GLM, specifically its inability to model non-constant and complex dispersion effectively. Deriving a human-interpretable expression, as illustrated in Eq. (H.7), proves challenging due to the inherent complexity involved in expressing the quantiles of a gamma distribution, even when we know the mean and dispersion. Therefore, we further employ Kernel SHAP decomposition to achieve an additive explanation for the DRN's prediction of the 90% quantile (Fig. H.14) from the ground up:

$$Q_{Y|\boldsymbol{X}}(0.9|\boldsymbol{x}^*;\boldsymbol{w},\boldsymbol{\beta}) \approx \underbrace{(+3.520)}_{\mathbb{E}_{\boldsymbol{X}}[Q_{Y|\boldsymbol{X}}(0.9|\boldsymbol{X};\boldsymbol{w},\boldsymbol{\beta})](\text{Figure H.14})} + \underbrace{(-0.048)}_{X_1\text{'s Contribution(Figure H.14)}} + \underbrace{(-0.108)}_{X_2\text{'s Contribution(Figure H.14)}}. \tag{H.9}$$

The close alignment between the estimated CDF by the DRN model and the true CDF, as showcased in Fig. H.14, highlights the model's effectiveness in accurately capturing various quantiles.

To address the global interpretability of the DRN, it's crucial to examine the contributions of X_1 and X_2 to the distributional properties across all instances. Mathematically speaking, a SHAP dependence plot repre-

sents the set of points $\left\{\left(x_{i,j}, \phi_j\left(v_{\mathcal{M}}^{\text{Mean}}, \boldsymbol{x}_i\right)\right)\right\}_{i=1}^{n}$ for $j \in \{1,2\}$. Fig. H.15 reveals that both $\phi_1(v_{\mathcal{M}}^{\text{Mean}}, \boldsymbol{X})$ and $\phi_2(v_{\mathcal{M}}^{\text{Mean}}, \boldsymbol{X})$ exhibit approximately linear dependencies on X_1 and X_2, respectively. Notably, $\phi_2(v_{\mathcal{M}}^{\text{Mean}}, \boldsymbol{X})$ does show a slight quadratic relationship as well.

H.2. Real dataset

This section examines the DRN's distributional interpretability from two key perspectives: local and global. These are demonstrated in H.2.1 and H.2.2, respectively. The DRN model referenced throughout the following subsections corresponds to the `DRN` reported in Table 3.

H.2.1. Local interpretability

For illustrative purposes, we focus on the policyholder with the second-largest mean adjustment in the test dataset. The left panel of Fig. H.16 displays the density functions predicted by the baseline and the DRN (solid lines), alongside the true observation (red dotted line) and the DRN's mean prediction (blue dotted line). The purpose is to decompose the DRN's mean prediction utilising Kernel SHAP values from scratch. In order of impact, this policyholder's top risk factors are `VehMaxSpeed`, `VehPrice`, `SocioCateg`, and `VehEnergy`. The magnitude of the SHAP values determines this ranking. Trust in the DRN may be enhanced by verifying that the explanations align with intuitive and expert expectations.

The right panel of Fig. H.16 demonstrates the corresponding mean adjustment decomposition. Comparing the left panel to the right panel, the almost identical significance of features and signs of the SHAP values reveals the baseline model's inability to fully capture the features' impacts on the mean prediction for this particular instance. Furthermore, the negative effects on the mean prediction attributed to vehicle price, exposure, vehicle energy and social category appear to have been overestimated or wrongly attributed by the baseline model.

In other cases, the refinements can be less severe, as demonstrated in Fig. H.17, where we plot the adjustments to two other examples:

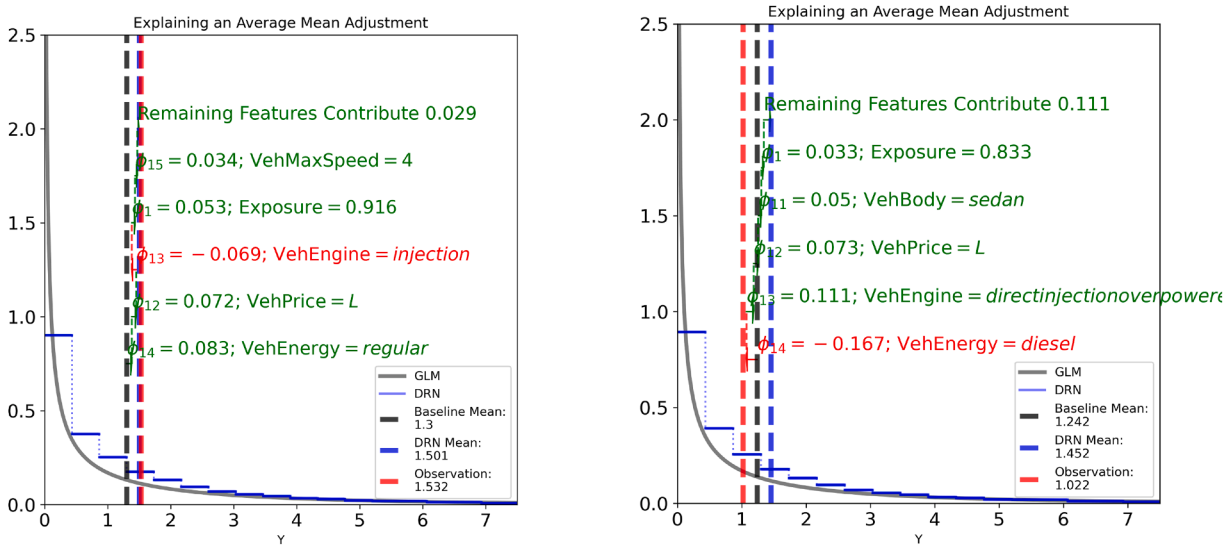


Fig. H.17. Two (different) examples of the refinements.

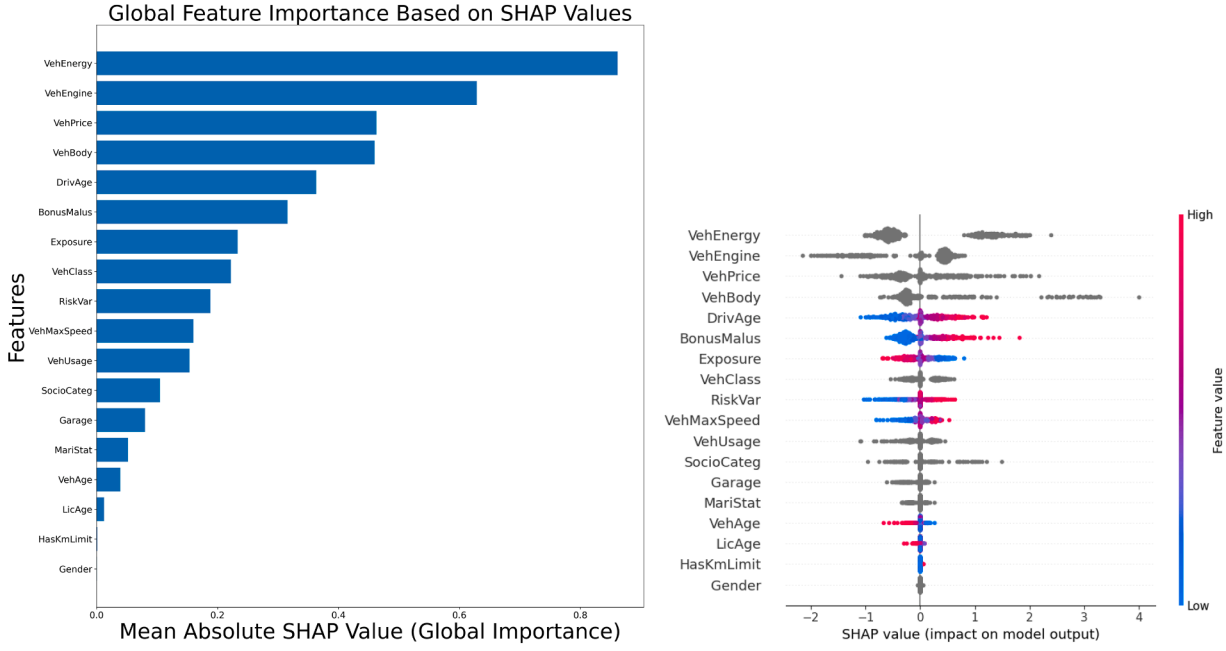


Fig. H.18. Left Panel: This displays the global importance of features in the DRN's mean prediction, ranked by their mean absolute SHAP values. Right Panel: A summary plot illustrating how these features impact the DRN's mean prediction.

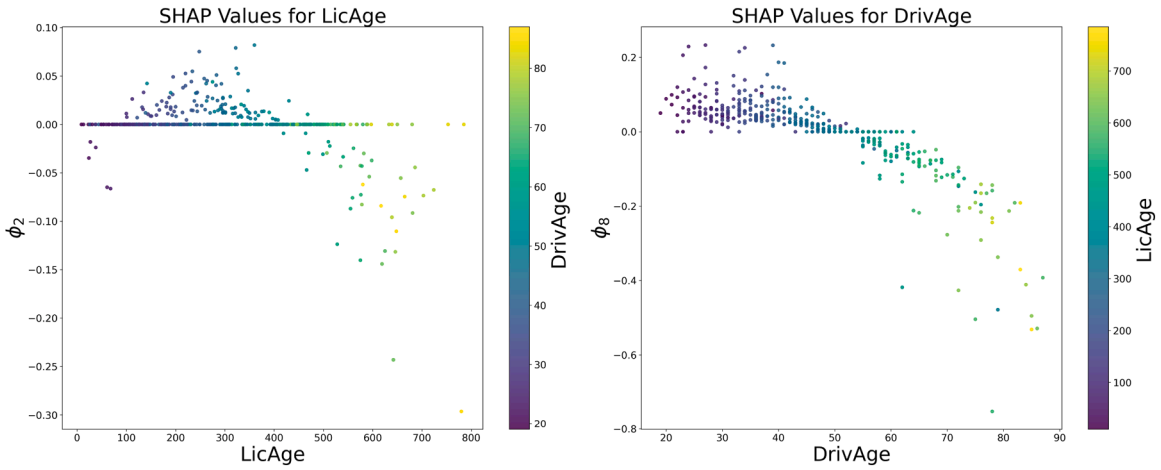


Fig. H.19. Mean adjustment SHAP dependence plot for `LicAge` and `DrivAge`.

H.2.2. Global interpretability

The left panel of Fig. H.18 highlights the global importance of features identified by the DRN while predicting the mean. It recognises `VehEnergy`, `VehEngine`, `VehBody`, `VehPrice` and `DrivAge` as the top five features for mean prediction in terms of the mean absolute SHAP values. The higher the mean absolute SHAP value, the more important the feature is. The right panel provides insight into how the features impact the DRN's mean prediction outcome. It is a beeswarm summary plot displaying the distributions of feature values and their global impacts. It reveals a positive association of the mean prediction with increasing `DrivAge`, `BonusMalus` and `RiskVar`. On the other hand, `Exposure`, `VehAge` and `LicAge` exhibit dual impacts: increasing the mean prediction at smaller values and decreasing it at larger values.

To explain the adjustments made by the DRN on a global level, we generate the so-called mean adjustment SHAP dependence plot, as demonstrated in Fig. H.19. Mathematically, we plot the points $\{(x_{i,j}, \phi_j(v^{\text{Mean}}_{\mathcal{M}_w} - v^{\text{Mean}}_{\mathcal{M}_\beta}))\}_{i=1}^n$ for $j \in \{\texttt{LicAge}, \texttt{DrivAge}\}$. A detailed analysis of the SHAP value distributions yields several interpretations and suggestions: i) incorporate a polynomial term in the GLM for `LicAge` or `DrivAge`; ii) explore interaction terms and possible dependencies between the features; iii) reevaluate feature importance based on mean absolute SHAP values, i.e., we can drop unimportant features during refinement to improve interpretability and reduce feature dimensionality if feasible. We refrain from implementing these possibilities in this paper while acknowledging the potential of these approaches to create a positive feedback loop between the baseline and neural network components.

Appendix I. Definition of Calibration

Definition 1 (Probabilistically Calibrated - Gneiting et al. (2007))**.** Let $(F_t)_{t=1,2,\dots}$ and $(G_t)_{t=1,2,\dots}$ denote sequences of continuous and strictly increasing CDFs, possibly depending on stochastic parameters. We think of $(G_t)_{t=1,2,\dots}$ as the true data generating process and of $(F_t)_{t=1,2,\dots}$ as the associated sequence of probabilistic forecasts. The sequence $(F_t)_{t=1,2,\dots}$ is probabilistically calibrated relative to the sequence $(G_t)_{t=1,2,\dots}$ if

$$\frac{1}{T}\sum_{t=1}^{T} G_t \circ F_t^{-1}(p) \to p \quad \text{for all} \quad p \in (0,1), \tag{I.1}$$

as $T \to \infty$.

References

Al-Mudafer, M. T., Avanzi, B., Taylor, G., Wong, B., 2022. Stochastic loss reserving with mixture density neural networks. Insurance 105, 144–174.

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., Farhan, L., 2021. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. J. Big Data 8, 1–74.

Bishop, C. M., 1994. Mixture density networks.

Burkart, N., Huber, M. F., 2021. A survey on the explainability of supervised machine learning. J. Artif. Intell. Res. 70, 245–317.

Corduneanu, A., Bishop, C., 2001. Variational bayesian model selection for mixture distribution. Artif. Intell. Stat. 18, 27–34.

Craven, P., Wahba, G., 1978. Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. Numer. Math. 31, 377–403.

Delong, Ł., Lindholm, M., Wüthrich, M. V., 2021. Gamma mixture density networks and their application to modelling insurance claim amounts. Insurance 101, 240–261.

Dong, T., Laub, P., 2024. drn: A Python Package for Distributional Refinement Networks. https://github.com/EricTianDong/drn.

Dunn, P. K., Smyth, G. K., 1996. Randomized quantile residuals. J. Comput. Graph. Stat. 5, 236–244.

Dutang, C., Charpentier, A., 2025. CASdatasets: Insurance Datasets. R Package Version 1.2-1. https://doi.org/10.57745/P0KHAG.

Embrechts, P., Wüthrich, M., 2022. Recent challenges in actuarial science, Annu. Rev. Stat. Appl. 9. https://doi.org/10.1146/annurev-statistics-040120-030244.

Fan, J., 1991. On the optimal rates of convergence for nonparametric deconvolution problems. Ann. Stat. , 1257–1272.

Fissler, T., Merz, M., Wüthrich, M. V., 2023. Deep quantile and deep composite triplet regression. Insurance .

Foresi, S., Peracchi, F., 1995. The conditional distribution of excess returns: an empirical analysis. J. Am. Stat. Assoc. 90, 451–466. http://www.jstor.org/stable/2291056.

Frees, E., Derrig, R., Meyers, G., 2014. Predictive modeling applications in actuarial science: Volume I: Predictive modeling techniques. https://doi.org/10.1017/CBO9781139342674.

Frees, E. W., 2009. Regression Modeling with Actuarial and Financial Applications. Cambridge University Press. International Series on Actuarial Science, https://doi.org/10.1017/CBO9780511814372.

Fung, T. C., Tzougas, G., Wüthrich, M. V., 2022. Mixture composite regression models with multi-type feature selection. North Am. Act. J., 1–33.

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. JMLR Workshop and Conference Proceedings. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, in: pp. 249–256,

Gneiting, T., Balabdaoui, F., Raftery, A. E., 2007. Probabilistic forecasts, calibration and sharpness. J. R. Stat. Soc. Ser. B 69, 243–268.

Gneiting, T., Raftery, A. E., 2007. Strictly proper scoring rules, prediction, and estimation. J. Am. Stat. Assoc. 102, 359–378.

Good, I. J., Gaskins, R. A., 1971. Nonparametric roughness penalties for probability densities. Biometrika 58, 255–277.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.

Jethani, N., Sudarshan, M., Covert, I., Lee, S. I., Ranganath, R., 2022. FastSHAP: real-time Shapley value estimation. ICLR 2022 .

Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Klein, N., 2024. Distributional regression for data analysis. Annu. Rev. Stat. Appl. 11.

Kneib, T., Silbersdorff, A., Säfken, B., 2021. Rage against the mean – a review of distributional regression approaches. Econometr. Stat. https://doi.org/10.1016/j.ecosta.2021.07.006.

Koenker, R., Bassett, , G.Jr, 1978. Regression quantiles. Econometrica, 33–50.

Kuleshov, V., Fenner, N., Ermon, S., 2018. Accurate uncertainties for deep learning using calibrated regression. PMLR. International Conference on Machine Learning, pp. 2796–2804.

Kullback, S., Leibler, R. A., 1951. On information and sufficiency. Ann. Math. Stat. 22, 79–86.
Laub, P. J., Pho, T., Wong, B., 2025. An interpretable deep learning model for general insurance pricing. arXiv preprint arXiv:2509.08467.
LeCun, Y., Bottou, L., Orr, G. B., Müller, K. R., 1998. Efficient BackProp. Springer Berlin Heidelberg.
Li, R., Reich, B. J., Bondell, H. D., 2021. Deep distribution regression. Comput. Stat. Data Anal. 159, 107203.
Lindholm, M., Wüthrich, M. V., 2024. The Balance Property in Insurance Pricing. Available at SSRN: https://ssrn.com/abstract=4925165 or https://doi.org/10.2139/ssrn.4925165.
Lundberg, S. M., Lee, S. I., 2017. A unified approach to interpreting model predictions. Adv. Neural Inf. Process. Syst. 30.
Makansi, O., Ilg, E., Cicek, O., Brox, T., 2019. Overcoming limitations of mixture density networks: a sampling and fitting framework for multimodal future prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7144–7153.
Marques-Silva, J., Huang, X., 2024. Explainability is not a game. Commun. ACM 67, 66–75. https://doi.org/10.1145/3635301.
Mayer, M., Meier, D., Wuthrich, M. V., 2023. SHAP For actuaries: explain any model. SSRN.
Molnar, C., 2020. Interpretable Machine Learning. Lulu.com.
Murphy, K. P., 2022. Probabilistic Machine Learning: An Introduction. MIT press.
Nelder, J. A., Wedderburn, R. W. M., 1972. Generalized linear models. J. R. Stat. Soc. Ser. A 135, 370–384. http://www.jstor.org/stable/2344614.
Ribeiro, M. T., Singh, S., Guestrin, C., 2016. "Why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144.
Richman, R., 2022. Mind the gap–safely incorporating deep learning models into the actuarial toolkit. Br. Act. J. 27, e21.
Richman, R., Wüthrich, M., 2022. LocalGLMnet: interpretable deep learning for tabular data. Scand. Actuar. J. 2023, 71–95. https://doi.org/10.1080/03461238.2022.2081816.
Rigby, R. A., Stasinopoulos, D. M., 2005. Generalized additive models for location, scale and shape. J. R. Stat. Soc. Ser. C 54, 507–554.
Romano, Y., Patterson, E., Candes, E., 2019. Conformalized quantile regression. Adv. Neural Inf. Process. Syst. 32.
Rossi, P. E., Allenby, G. M., 2003. Bayesian statistics and marketing. Market. Sci. 22, 304–328.
Rügamer, D., Kolb, C., Klein, N., 2023. Semi-structured distributional regression. Am. Stat., 1–12.
Schelldorfer, J., Wüthrich, M. V., 2019. Nesting classical actuarial models into neural networks. SSRN.
Silverman, B. W., 1985. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. J. R. Stat. Soc. Ser. B 47, 1–21.
Srivastava, R. K., Greff, K., Schmidhuber, J., 2015. Highway networks. arXiv preprint arXiv:1505.00387.
Taylor, J. W., 2000. A quantile regression neural network approach to estimating the conditional density of multiperiod returns. J. Forecast. 19, 299–311.
Thielmann, A. F., Kruse, R. M., Kneib, T., Säfken, B., 2024. Neural additive models for location scale and shape: A framework for interpretable neural regression beyond the mean. PMLR. International Conference on Artificial Intelligence and Statistics, 1783–1791.
Tran, M. N., Nguyen, N., Nott, D., Kohn, R., 2020. Bayesian deep net GLM and GLMM. J. Comput. Graph. Stat. 29, 97–113.
Wang, Y., Oka, T., Zhu, D., 2023. Bivariate distribution regression with application to insurance data. Insurance 113, 215–232.
Wegkamp, M., Yuan, M., 2011. Support vector machines with a reject option. Bernoulli 17. https://doi.org/10.3150/10-BEJ320.
Wilcoxon, F., 1945. Individual comparisons by ranking methods. Biometr. Bull. 1, 80–83. http://www.jstor.org/stable/3001968.
Wüthrich, M., 2022. The balance property in neural network modelling. Stat. Theory Rel. Fields 6, 1–9. https://doi.org/10.1080/24754269.2021.1877960.
Zeng, D., Lin, D., 2007. Maximum likelihood estimation in semiparametric regression models with censored data. J. R. Stat. Soc. Ser. B 69, 507–564.